Big Data Analytics for PV Systems Real-time Monitoring

Lu Liu

Thesis submitted for the degree of
Master in Informatics: programming and networks
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2018

Big Data Analytics for PV Systems Real-time Monitoring

Lu Liu

2nd May 2018

Big Data Analytics for PV Systems Real-time Monitoring

Lu Liu

May 2, 2018

Abstract

Solar energy is one of the most influenceable renewable resources. Photovoltaic(PV) system is widely used in converting solar energy into electricity. Therefore, monitoring the working state of PV systems in real-time to make sure the PV systems working in reasonable condition is a crucial task. The key work in this task is forecasting PV power generation in real-time. Since the PV power generation is greatly depend on the weather conditions which include many variables. This problem becomes a big data issue. Techniques which are relevant with big data involving data mining, machine leaning and deep learning are adopted in solving this problem. A solution is given by utilizing these techniques. Various methods of data mining are adopted in analyzing the big data, and machine learning and deep learning algorithm are utilized in implementing five forecasting models for PV power generation.The five forecasting model are : near regression model, Lasso regression model and Ridge regression model, SVR model, and MLP model. In order to have good view of the forecasting results, visualization corresponding with the five models are given. The five forecasting model are tested and evaluated by means of many different measurement including explained variance score (EVS) mean square error(MSE), and R^2 score(R2) and processing time. The evaluation result of the five forecasting model is reasonable which can be taken into utilization in PV system real-time monitoring.

Key words: big data, machine learning, Photovoltaic(PV), forecasting, real-time monitoring

Contents

1 Introduction

1.1 photovoltaic(pv) system

Studies of renewable resources is becoming a significant topic in order to solve the problem of fuel decrease and environmental issues such as air pollution and green house. Solar energy as a type of green, free and inexhausted energy is considered as one of the most important renewable energy to meet the increasing energy demand of the world.

Photovoltaic (PV) is one of the equipment utilized to convert sunlight directly into electricity. A solar PV system is powered by many crystalline or thin film PV modules, and PV module is composed of numbers of solar cells which are generally made from light-sensitive semiconductor materials such as silicon that use photons to dislodge electrons to drive an electric current.[1] PV solar cells are interconnected to form a PV module to capture the sun rays and convert solar energy into electricity. So when the PV modules are exposed to sunlight, they generate direct current. It is one of the best way in nowadays which can transfer the solar energy into utilization. Many countries in the world have taken this technique into utilization, however, the estimation of the PV generation is a challenge because the PV system generation is greatly affected by the weather conditions.

1.2 motivation

Solar energy as a green renewable energy is starting to show the potential in sharing of the global electricity production. Many industrialized countries have installed significant PV systems to supplement an alternative to conventional energy sources while an increasing number of less developed countries start to adopt solar to reduce dependency of expensive imported solar.

In the world, some Asian countries have expressed their intention to continue to develop PV system. For the North America the market remains steady and it could continue to increase the power generation at a reasonable speed. In Europe, the PV market will be more competitive instead of financially supported currently. All these situation in the word should maintain that the PV market is growing fast and vital in the coming years. [2]

However, the PV generation is not stable, and is strongly depend on the weather condition. Thus, the generation of PV system is not controllable. Except that, the key modules is getting degraded with the time of PV system working increasing, and sometimes the PV panel may get damaged by other factors such as the extreme weather or by creatures.

In that situation, the amount of PV generation is far deviated from it is expected. It is harmful for the utilization of PV power.

Thus, it is significant to diagnose the working condition of PV systems and monitoring PV systems in real-time to keep PV system work in a condition which is expected. Once there is a problem it should be detected in time so thus better decision can be made to avoid risk. For example, detecting equipment failure and identify key modules degradation pattern are crucial for the health of PV system.

The key challenge to accomplish this work is forecasting PV generation in real-time.

1.3 goal

As the weather has great influence on the production of PV systems, such as irradiation, temperature, humidity, wind velocity. The goal of this thesis is to build the relationship between the weather with the production of PV systems by analyzing history data. Through the model, we are able to use predicted data of weather in the near future to predict the production of PV systems. Once the result deviate much from prediction, probably there are problems with PV system, and the reason need to be figured out, and then adopt proper measure to fix the PV system and make better decisions. For example, according to the accurate forecast, PV system operators can balance the consumption of power and reserve spared power for emergency.

In order to predict the generation of PV system based on weather conditions, huge amount of history data of PV generation and the corresponding weather data with a variety variables at the same location in the same time period is required. Since it is a big data issue, some relevant techniques should be taken into consideration in solving this big data problem, such as data mining, machine learning, deep learning. In addition, tools such as Python, numpy library, scikit learn library,matplot library would also be used in analyzing data and building the forecasting model.

1.4 Structure of thesis

There are 9 sections in this thesis. In section 2, the back ground knowledge and relevant study is presented. In section 3, there will be a briefly introduction about the techniques which is closely related to the research. In section 4, detailed explanation about data analytics methods will be given. In section 5, the part will give the implementation of the five different PV generation forecast model, while section 6 will give a evaluation for the five PV forecast model. In section 7, some work related with real-time monitoring will be described. Section 8 will give a conclusion to the work in this thesis. In section 9, a discussion based on the task accomplished will be given and future research direction will be mentioned.

2 Background

2.1 Related research

Since there are various meteorological factors affect PV generation, it is difficult to predict and control the PV generation. Many researchers made effort in this area and much technologies has been adopted to forecast PV power generation. Some Technologies, such as satellite cloud, numeric weather prediction(NWP), have been devoted in PV power generation forecasting. The accuracy is in the rang of 80%-90% for short term PV power generation. However, in some extreme weather condition like rainy, snowy or in the morning, in the evening, the forecasting error rate is high, and sometimes the relative mean square error (RMSE) can be higher than 50%.[3]

Many researchers adopt a machine learning algorithm and build a model to forecast the PV power generation. In general, there are three types of methods in forecasting PV power generation: The first one is to build a forecasting model directly. The second one is to classify the the history weather data and PV power generation data into several types based on the weather conditions and build the corresponding model according to each type.

[4]proposed a mathematical model to predict the production of PV panels using SOFM neural model. The SOFM neural model is based on unsupervised feedback neural networks. A sensitive analysis is performed in building the forecasting model.The data for this research is from Oman which is a quite hot city, it is well know that the temperature and the radiation play an crucial role in PV power generation. In that city, both the temperature and radiation are in a high level.So an sensitive analysis is performed to get the sensitive value of the temperature and solar radiation. The conclusion is that the radiation plays an far more important role than temperature. The solar radiation data was measured between July 2013 and August 2014. The data set was divided into three set. 40% of the data set was used for training, 20% of the data set was used for cross validation, and the rest 40% of the data set was used for testing the output of the network. This paper also made a comparison between this proposed model with a SVM model and a MLP model. It indicates that the performance of the SOFM neural model performs better than others.

[5] adopts self-organized map(SOM) to classify the local weather type of 24 h ahead provided by the on-line meteorological service. The type of weather is classified in three groups by this algorithm: sunny, cloudy and rainy. Based on these three group, the overall system consists of three models: the sunny photovoltaic power generation forecasting

model, the cloudy photovoltaic power generation forecasting model and rainy (snowy) photovoltaic power generation forecasting model. After building the three models, the radial basis functions (RBF) is used for developing an on-line short-term forecasting model of 24 hours ahead photovoltaic system power generation. The parameters of the input values of the model are: the mean daily solar irradiance, the daily air temperature, the mean daily relative humidity, the mean daily wind speed and the mean daily power output of the PV system. The output is the 24 hours ahead of power generation of the PV system. The result shows that this method is acceptable, but it is worth noting that the testing data scale is a bit small and the accuracy of the rainy (snowy) photovoltaic power generation forecasting model is quite low.

[3] Proposed that there is a high correlation between power generation from the same weather type every day. Based on this, history power generation are divided into four types which are clear sky, rainy day, cloudy day and foggy day according to weather conditions. In each group, the data sample are divided into two groups, training data and testing data. For each group, support vector machine(SVM) is used in establishing the one day-ahead forecasting model , and the RBF function is selected as the kernel function. In the testing phases, one of the four models is selected based on the forecast weather of the next day. The input data set of each model include 15-min-interval historical PV power from the nearest day of the same weather type and weather data(include the maximum temperature,the minimum temperature and the average temperature) from the local weather report. The output of the model is the PV power generation forecasting result of the next day with a 15-min interval.The authors of this paper pointed that SVM has better calculating speed and good convergence result compared with ANN, and SVM will not be trapped into local minimum values. In this paper, the method in classifying the weather type is not described.

[6]Proposed a prediction model of PV system for Oman. The data use for prediction are collected from an installed system in location called Sohar. SVM is also used in establishing the prediction model in this paper. But there is not a classification for weather type before the regression. The two inputs of the prediction model are solar radiation and ambient temperature. The output of the prediction model is PV current.

[7] indicated that the AI(Aerosol Index) has a strong linear regression with solar radiation(conventional models have taken into consideration the humidity,temperature, and wind speed), so it has the potential influence to the PV power generation. A linear regression analysis between AI and PV power was performed b using Pearson conduct-moment correlation. The data used for this research was a two month data set from

April 1, 2013 to May 31, 2013 from Gansu, China. Base on this, a forecasting model with the utilization of back propagation (BP) is proposed. With the considering of AI data as an additional input parameters, this model implemented 24-hours ahead forecasting which performs pretty good result.

[8] designed two model for PV power prediction in Oman using SVM and Multilayer Perceptron (MLP) respectively. The SVM and MLP models are consist of two input layers and one output layers. The inputs of the SVM model are time and solar radiation, the output of the SVM model is the PV current, while the inputs of the MLP model are ambient temperature and solar radiation. The two models have been evaluated on the basis of Mean Square Error (MSE). In comparison of MSE between the SVM and MLP models, result shows that MLP model is more accurate than the SVM model.

[9] Proposed four neural computing techniques for simulating and predicting the power generation of solar energy system. The four models are SVM, MLP, self-organization feature maps (SOFM) and generalized feed forward networks (GFF) . These four models are compare in terms of MSE, MAE, NMSE,accuracy and R. The input of the four models are solar radiation and ambient temperature from Sohar in the period of two years. The output of the models are the PV array voltage and the current. The experiment data are from a PV system installed in Sohar, Oman for a period of one year. The result of the comparison shows that GFF model yielded a highest accuracy among the four models, while SOFM gave a smaller MSE value compared with the other three models. In terms of NMSE, all the four models scored good result, especially the SVM model achieved a value of 0.0039.

[10] Present a global solar irradiation forecasting method in a indirected way. Feed forward multilayer perception algorithm is adopted in developing the artificial neural networks(ANNs) model. The inputs of this model are latitude, longitude, day number and sunshine ratio; The output of this model is the clearness index . The clearness index is used to calculate global solar irradiation. In this paper, research data are from 28 weather stations of Malaysia, 23 stations' data were used to train the network and 5 stations' data were used to test the model.

[11] performed an estimation of solar radiation by using artificial neural networks(ANN). Resilient propagation(RP) and Scale conjugate gradient(SCG) learning algorithm is used. The transfer function adopted in this model is logistic sigmoid. The meteorological data used in the research was in a period from August 1997 to December 1997 from 12 cities spread over Turkey. 9 stations of the data was used for training and 3stations of the data was used for testing. A three layer ANN is used in

building the forecasting model, the input layer include both meteorological data an geographical data which include 6 features. They are mean diffuse radiation and mean beam radiation,latitude, longitude,altitude and month. The output of the model is the solar radiation. For training and testing the forecasting model, the solar radiation has been estimated as monthly mean daily sum by taking the use of Meteosat-6 satellite C3 D data in a visible range over the 12 cities in Turkey.

[12] proposed two models for PV generation production forecasting. On is based on artificial neural networks(ANN) and another is based on multiple linear regression(MLR), both of the two models are build in 1-minutes time step. The author used 5 different settings for the input variable in order to get the most influencing variables for PV power prediction. The variables are: solar radiation, air temperature, back surface module temperature, open circuit voltage and wind speed. After the 5 testing of the different combination of setting input variables of the two model, the result comes at the solar radiation and air temperature are the two most influencing variable. In building the ANN forecasting model, one hidden layer is used, and the formulation adopted by calculation the numbers of nodes in hidden layer is:$H_n = \frac{I_n+O_n}{2} + \sqrt{S_n}$, H_n is the estimated number of nodes in hidden layer, I_n is the number of features of input layer, O_n is the number of variables in output layer S_n is the number of data samples. In addition, by comparing the result of the forecasting of the ANN model and MLR model by taking the measurement of mean absolute percentage error(MAPE), the result shows that the MAPE of the ANN model is smaller than the MLR model, thus the author gave the conclusion that the prediction of ANN is better than MLR model.

[13] Presented an artificial neural network(ANN) model for estimating monthly mean daily diffuse solar radiation. .The solar radiation data used for training and testing for are collected from 9 cities with different climate conditions all over China during 1995-2004. The data from 8 cities are used for training process and data from one city are used for testing. The feed-forward back-propagation algorithm with single hidden layer is used in building this forecasting model. The input variables of the model are monthly mean daily clearness index, sunshine percentage. The output of the model is monthly mean daily diffuse fraction. In terms of choosing the number of nodes in the hidden layer, the formulation used is:$m = (p+q)^{0.5} + \alpha$, m is the estimated number of nodes in hidden layer, p in the number of variables in th input layer and q is the number of variables in the output layer. According to the test in the research, the best number of nodes in the hidden layer is 5. In order to evaluate the performance of the ANN model, estimated values are compared with

the measured values with three measurements adopted,mean percentage error(MPE), mean bias error(MBE) and root mean square error(RMSE). Based on the evaluation result, the author give the conclusion that the estimations for solar radiation by ANN are in good agreement with the actual values and are superior to other empirical regression models.

[14] presented a high level look at some of the tools available in Matlab tool set which helps the user to extract information from 'big data' source. The author in this paper implemented a prediction of the amount of solar power generation by a micro-grid. The data used for the research was from different source and combined together. Part of the data was from the Solar Radiation Research Laboratory(SRRL), and some are collected from Measurement and Instrumentation Data center(MIDC). There are approximately 250 different variables after the data combination. Before building the forecasting model, some data pre-processing methods are adopted. First is cleanup of the raw data. Data points containing invalid values are removed from the data set. Second one is nonlinear data set expansion, and principle component analysis(PCA) was also performed. After the data pre-processing , two non-parametric model generation tools are used in building the model. The fuzzy Inference system Generator and Back-propagation Neural Network training tools. For the dimensional reduction task, except PCA, genetic algorithm were used to reduce the dimension of the data set. In order to generate the best non-parametric model,different combination of data variables are performed. The conclusion of the testing was that the best model that predicting solar radiation was the one using the maximum number of original variables and then reduced by utilizing PCA. The author also gave the conclusion that the non-parametric model generation methods performed significant better than s sub-optimal predictor. except that, the model utilizing Back-propagation Neural Network training tool provides significant better training time, especially when the dimension of the data set is high.

[15] implemented two ANN models for estimating solar radiation by first estimating the clearness index. One is Radial Basis Function(RBF), and the other is Multiple Layer Perceptron(MLP), The data for investigating was collected from eight station in Oman. The RBF network implemented was a three layer network, the activation function in the hidden layer is Gaussian. While the MLP network adopted sigmoid function. There are five variables for the input layer for both RBF and MLP network. They are Latitude, Longitude, Altitude, Sunshine duration and solar radiation. According to the evaluation result of the two forecasting models by utilizing root mean square error(RMSE) measurement, both of the two models performs good in modeling the data set,while, the RBF

model requires less computing time.

[16] Present a study to predict daily global solar(DGS) radiation by adopting ANN techniques. The data used for the research was from the period 2002 to 2006 for Dezful city in Iran. The meteorological variables include daily mean air temperature,relative humidity, sunshine hours,evaporation and wind speed. In order to see the effect of the six features, six different combinations are formed. MLP and RBF networks are applied in building the forecasting model, then the two models are performed by utilizing the six combination of the variables. The measured data from 2004 to 2005 are used as training data, and the data of 214 days in 2006 are used as testing data. In evaluating the performance of the two models, mean absolute percentage error(MAPE) is adopted as the measurement. The best result of the MLP models obtained 5.21% which shows a good result.

[17] proposed that accurate prediction of meteorological variables is crucial in forecasting the generation of renewable energy such as wind and solar resource. It is a big data problem for accurate prediction of these meteorological variables which requires a multiple of disparate data, multiple models and application of computational intelligence techniques to blend all of the model and observational information in real-time. In explaining this, a case study of the National Center for Atmospheric Research (NCAR) is provided, called SunCast Solar Power System. Basically, SunCast Solar Power System is constructed by two systems. The NWP forecast and Nowcast System. The SunCast system trains a model regression tree by taking use of the measured irradiance value and the power measurement. The model requires historical data for training and testing, then the model is applied in real-time utilizing the irradiance forecast to produce a power forecast. In the case of SunCast Solar Power System,data mining technique is used for blend models and complex observations. Machine learning technique is also adopted in building the forecasting system.

[18] Points that the dynamic energy management in smart grid is a big data issue, the challenge of this issue trying to take advantage of the users' participation in order to reduce the cost of power, and accurate forecasting of the renewable production is critical in implementation a dynamic energy management. Thus, this requires intelligent methods and solutions to explore large volumes of data generated by renewable energy resource ,smart meters and users. Hence, this calls for some state-of-the-art techniques such as robust data analytics, high performance computing, cloud computing techniques and efficient data network management in optimizing the operation of smart grid. The author provided a brief description of some most commonly used data

processing methods in literature. In terms of data mining and predictive analytics in SGS, the author gave four methods including dimensionality reduction, load classification, short-term forecasting and distributed data mining. For high-performance computing, dedicated computational grid and cloud data are proposed. As for future direction, some methods are mentioned in building an accurate real-time monitoring and forecasting system, such as feature selection and extraction, on-line learning, randomized model averaging, mapreduce parallel processing and available tests and platforms.

3 Related Technology

3.1 Big data

Big data is a term used to identify datasets that are in huge volume and high complexity that we can not manage with tradition methods or data mining technology software tools such as a relation based database. In order to describe the characteristics of big data, 5 Vs are usually used. They are Volume, Velocity, Variety,Veracity and value.

•*Volume* refers to that the data volume is vast, and the size is continues increasing, so that the traditional tools can not process. With the help of distributed system, huge volume of data could be stored separately and combined together

•*Velocity* refers to that the speed at which new data is generated and the speed at which data moves around. [19], and we expect to get the useful information from it in real-time. The data set grows rapidly is partly because the development of Internet of things such as mobile device, remote sensor, wireless sensor, even renewable energy system.

•*Variety* indicates that the data type for the data set is complex, there different types of data in one data set, such as text, audio, video, graph and so on.

•*Veracity* refers to the trustworthiness of the data, with the variety forms of data, and vast amount of data, the quality and accuracy of data is not easy to control. However, the volumes usually can make up for the lack of quality or accuracy of part data.

•*value* refers to the business value that can provides organizations a competitive advantage, due to the ability of collection and leveraging big data.[20]

In the case of analyzing the relation ship between PV generation and weather variables and making prediction for PV generation is a big data problem. The data volume for the PV generation and weather variables is huge and the speed of generating new data is fast. Take the example of one PV system in Institute for Energy Technology(IFE), the PV power generation is recorded in 15 min-interval, and the data is stored in a period of several years. As for the weather data, there are a large size of history weather data recorded with hundreds of parameters. Thus, exploring the value among the data by adopting some analytics methods is a big data issue.

Recently, the term 'big data' tends to refer to the use of predictive analytics, user behavior analytics or some other advanced data analytics methods that extract valuable information form big data set, and seldom refer to a particular size of data set.

Dealing with big data problem have many challenges including data capturing, data collecting, data storage, data cleansing, data analysis, data visualization, updating, and information privacy.

There are two main strategies for dealing with the huge volume of big data. First one is sampling, sampling is refer to that the volume of data is too big,we can't use the full set of the data, so we obtain a subset of the data set by adopting some methods. Second is using a distributed system such as Apache Hadoop. [21]

Apache Hadoop is an open source software framework for storing, processing data and running applications on clusters of computers. Hadoop provides massive storage for almost any type of data. Hadoop framework breaks big data into blocks and store them on clusters of commodity hardware. It also provides high processing power and ability to handle huge amount of tasks or jobs simultaneously by using multiple low-cost computers for fast results. Currently, two core modules in the basic framework of Hadoop is Hadoop Distributed File System (HDFS) and Map Reduce.

HDFS framework can accept data from many sources, both structured and unstructured. In HDFS framework, there are two cluster, Name Node and Data Node. The aim of this is to separate the task of management and computing. Name Node is in charge of working state management and record the situation of machine, while Data Node play the role as computing and executing. Name Node and Data Node cooperate with each other. Data divided in blocks with the same size is stored on some Data Nodes. Data Node send the message of its running state and content stored to Name Node and execute as Name Node instructed. Name Node receive a user's request and then send the the location of the content stored back the user. Afterwards, the client communicate with Data Node directly and get the result of the computing and operating on the data.The process of data stored in HDFS is shown in figure below:

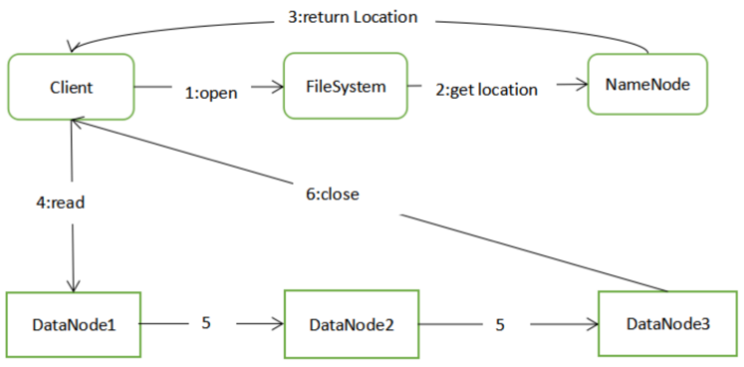


Figure 1: Data stored in HDFS

MapReduce is a software programming model for processing large amount of data distributed on a cluster of computers in parallel[22] . As the name implies, MapReduce consists of two methods map() and reduce(). Map() methods get task from Split which is stored in Block,and then it groups data according to a key, map() method run each data entity on a number of processes in parallel and then passes them to the reduce() method. The reduce() method will process each group and merges these values together with the same key to form a smaller set of values. the reduce() method process each group also in parallel. After the processing , it return a collection of values. MapReduce is a framework which is applicable in some lots of different ways, it can process both structured data and unstructured data. This process is described as figure below:

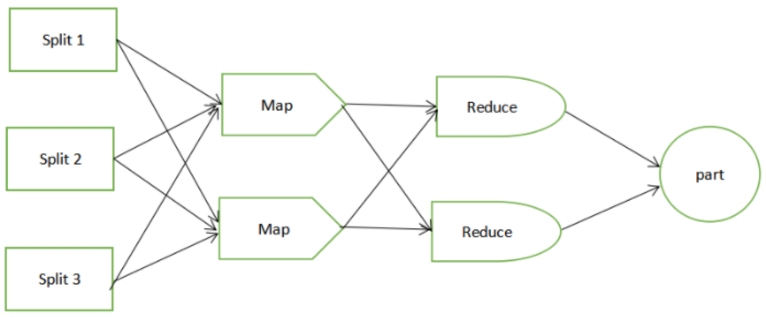


Figure 2: Process of MapReduce

In addition, with the rapidly increasing of big data, some techniques such as data mining and machine learning, deep learning are all play significant roles in solving big data issue.

3.2 Machine learning

Machine learning is one of the technique which can be utilized in solving big data issue. It uses statistics techniques to make computer system having the ability to learn and predict from the given data. The goal of machine learning is to learn some properties of a data set and then apply them in new data. Thus, a data set is usually divided into training set and testing set. The training set is used for learning data properties while the testing set is used for testing the properties and evaluating the performance of algorithm.

Actually, machine learning is a field which is a combination of many other subject, firstly, machine learning is closely related to computational statistics which focus on making prediction by using statistics analysis.

It is also strongly tied with mathematical optimization, which provides methods, theory and application domains to this field. Many machine learning problem is solved by minimizing the loss function, which is used to represent the discrepancy between the predictions of the model and the actual values. The difference between machine learning and mathematical optimization arises from the goal of generalization: the mathematical optimization aims at minimizing the loss on the training set and machine learning is focus on minimizing the loss on unseen samples[23].

Sometimes, machine learning is also overlapped with the data mining technique. They all give solutions based on experience data or history data. However, data mining focuses on discovery the unknown properties of the data, while machine learning focuses on prediction of the new data based on the training of old data.

Machine learning task are usually classified into two broad categories, according to if there is a label of feedback available in a learning system. They are supervised learning and unsupervised learning. If the data samples which the machine learning algorithm is trained on include input signal and output signal, (or the features and target). Then the goal in the machine learning task is learning the rule which can map inputs to outputs; If there is no label is given for the learning algorithm, the algorithm should find the structure of the training data set on its own.

In addition, for some cases, only part of the out put signal is given for the training data set. This is classified into the type 'Semi-supervised learning'. In this kind of task, the training data set is incomplete with some output signal missed. 'Active learning' is another type of task in machine learning, the output labels in training data set is limited, and it allows interaction with users to provide some information about label.In 'Reinforcement learning', there is no output signal like in supervised learning, but a feed back (often in form of reward or punishment) provided based on the ongoing situation, such as driving a vehicle or playing a game against an opponent[24].

When the desired output of a machine learning is considered, machine learning task could be classified into several types, the first one is 'classification', the goal of this type of task is classifying the data into two or multiple classes according to the output label of the input data. It is a typically tackled in a supervised way. An example is face image recognition, if the aim is dividing the image into two or several groups, it is a classification problem. The second is 'regression', in this kind of task, the outputs are continuous value rather than discrete value. it is also supervised problem. For example, the task of predicting the price of stock is a problem of regression type. The third is clustering, there is no output signal in training dataset, the goal of this kind of task is divide inputs into groups, but there is no direction directly as classification problem. The number of groups nor how to divide is not known in advance, the algorithm should find the rule by its own. So it is a typical unsupervised learning problem. The forth is density estimation, the aim of this type of task is to determine the distribution of data within the input space. It is a unsupervised learning. The fifth is feature selection, it aims at projecting data from high dimension to lower dimension. It is also an unsupervised learning type.

The relationship of the classification of machine learning types are shown in the figure below:

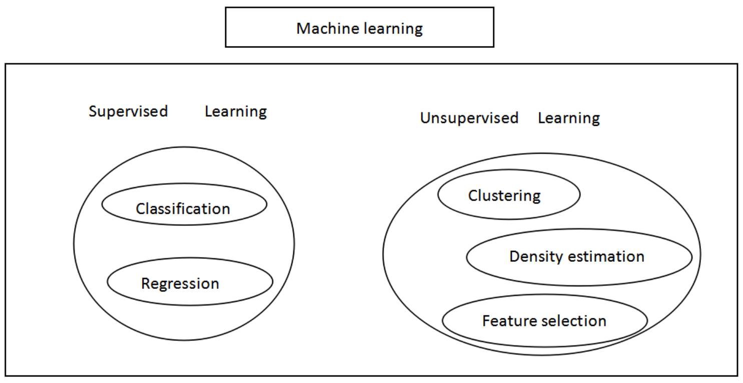


Figure 3: Relationship among classification of machine learning types

The problem of PV generation forecasting in this research is a regression problem, because the desired outputs are continuous value.

3.3 Data mining

Data mining is another techniques in dealing with big data issues. In big data analytics, data mining is a process of discovering patterns in big data set by adopting methods which is similar with machine learning, statistics, and database systems[25]. It is a crucial process in dealing with big data. The goal of data mining process is discovering properties and extracting information from data set, and transform it into data structure which is understandable and could be used in future[25]. Data mining is the analysis step of 'knowledge discovery in databases' process, or KDD. The process of Knowledge discovery in database(KDD) process could be defined with the steps[26]: • Data collection

- Feature selection
- Pre-processing

- Transformation
- Data mining
- interpretation/evaluation.

Before data mining is performed, feature selection and pre-processing are required. For data collection, a common source for data collection is a data mart or data warehouse. The aim of feature selection is distinguishing the importance of features for target data. As for pre-processing, it is essential in analyzing the data before data mining, it aims at cleaning the false data and missing data. The data mining step aims at discovery the patterns in the data set by adopting some algorithms,The algorithms which are efficient and widely used includes: Memory-based reasoning; Market basket analysis; decision trees; generic algorithm; cluster detection; link analysis; on-line analytic processing; neural networks; discriminant analysis;logistic analysis. The last step evaluation is used for verify the discovery in data mining algorithms in another data set. Sometimes the patterns found in the training data set will not be present in the general data set, this is called over-fitting. To overcome this problem, in the evaluation step, a testing set is used. A comparison is performed between the desired output with the actual output. Then the accuracy could be measured by the comparison.

Tasks are divided into 6 classes commonly in data mining[26]: • Anomaly detection, the aim of this task is distinguishing unusual data records, which should be further investigated.

• Association rule learning, in this type of task, the goal is analyzing the relationships between variables. For an on-line shopping website, it probably needs to acquire the searching history of some products in order to discover which products are usually searched at the same time period by a single user. Then these products may have close relationships, they could be put links on the one else. this kind of task often adopts the method 'market basket analysis'.

• Clustering, the aim of clustering task is discovering the groups in the data according to the similarity of them without knowing the number of groups or the structure of the data in advance.

• Regression, this type of task attempts to deal with data by learning a function that maps a data item into a real-valued prediction variable. it aims at estimating the relationship among data with the least error.

• Summarization, the type of task involves finding a compact description of a subset of data, such as report or visualization.

Some of the methods adopted in data mining is similar with machine learning. However, there is a little difference in the goal of data mining and machine learning. Data mining focus on discovering the properties of the data set, while machine learning emphasis on learning the properties from the data set and making prediction for new data based on the learning result.

Data mining tools can be divided into too main classes[27]: model building and pattern discovery. Model building is giving summary of data set from a high level and global view which include modern statistics methods such as:regression models, cluster decomposition and Bayesian networks. the model building usually gives a overall description of the data set. However, the pattern discovery focus on the local structure of data set, the patterns are often embedded in a mass of irrelevant data[28].

Data quality is a basic issue in pattern discovery[27]. When discovering work is performed in million of data, manual checking is not a practical way, thus, data cleaning and imputation procedure requires a automatic method. However, there is a risk in automatic cleaning data. Because this is possibly dismissing the interest or erase or smooth the characters.

3.4 Tools

This part gives a explanation about the tools used in this research. They are Python language, scikit learn library, numpy package and matplotlib, which are all employed in implementing forecasting models.

3.4.1 Python language

Python is a simple and efficient programming language with high-level data structures. The python interpreter and extensive standard library are available for free in source or binary form for all major platforms from the Python Web site.

The Python interpreter is easy to be extended with new functions implemented in C language or C++ language (or other languages callable from C), and data types can also be extended in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications[29].

3.4.2 pandas library

Pandas is an open source python data analysis library. It provides high-performance, flexible data structures and data analysis tools for Python programming language[30].

The data structures in pandas is fast, flexible, expressive and easy to use. the data structures designed make it easy and intuitive when working with "relational" or "labeled" data[31].

Pandas is good at dealing with many different kinds of data: • Tabular data with heterogeneously-typed columns, such as in Excel spreadsheet or in an SQL table.

• Time series data, ordered or unordered which means not in fixed frequency.

• Matrix data with arbitrary rows and columns.

• Any other form of observation data set or statistical data set.The data even not need to be labeled to be placed into a pandas data structure.

Two primary data structures in Pandas are 'Series' and 'Data Frame', 'Series' handles with 1dimension data while 'Data Frame' handles 2-dimension data. For R users, 'Data Frame' provides all the functions in R's data.frame, and except that, more functions are added. Pandas is built based on Numpy, and it aims at integrating with many other third party libraries within a specific computing environment. A large number of typical use case in statistics, finance, social science and many area in engineering are handled with Pandas.

3.4.3 numpy package

Numpy is a fundamental package which provides scientific computing with Python. It includes a powerful N-dimensional array object, sophisticated (broadcasting) function library, and tools used for integrating C, C++, and fortran code, it also provides efficient linear algebra, Fourier transform, and function for generating random number. Besides its obvious scientific uses, NumPy can also be utilized as an efficient multi-dimensional container of generic data[32].

3.4.4 scikit-learn library

Scikit-learn is a efficient tool used for data mining and data analysis. It provides various machine learning algorithms in Python, based on Numpy, SciPy and matplotlib. It is accessible to everyone and reusable in various contexts. It is open source tool with BSD license and can bu used for commercial. In scikit-learn, many machine learning algorithms has been implemented with different types task solved including classification, regression, clustering. In addition, there are also some data mining methods implementation in scikit-learn package.

3.4.5 matplotlib library

Matplotlib is a Python 2D plotting library. It is a great plotting tool which can produce publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Except that, matplotlib can be used in Python scripts, the Python and Ipython shells, web application severs and Jupyter notebook. By adopting matplotlib package, many types of figures can be generated. such as plots, histograms, bar, chart, errorcharts, power spectra, scatterplots. It is an efficient tool for plotting and visualization. Thus, it is widely used in data analysis.

4 Data analysis

4.1 data collecting

In analyzing the correlation between weather variables and PV power generation and monitor the working state of the PV system, a huge amount of PV power generation data and the corresponding weather data is required. In order to do a big data analytics, the measure should in a long period. The Institute for Energy Technology(IFE) which is located in Oslo, Norway has installed many PV systems include ground panels and wall panels. The recorded data of PV generation for some PV systems in recent years are available. So the data of PV power generation used for research in this paper is from IFE. The PV generation data is produced by a ground panel in IFE located at Milaveien, Kjeller, Oslo. It shown in the figure below:

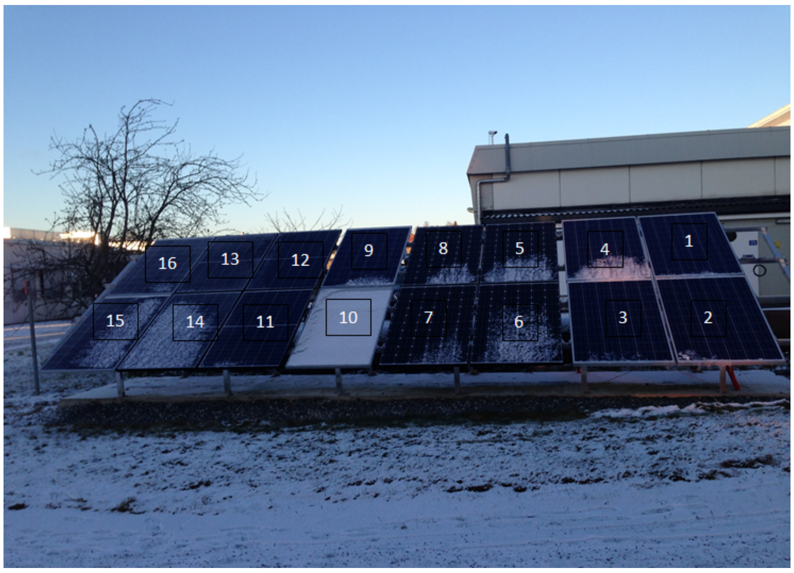
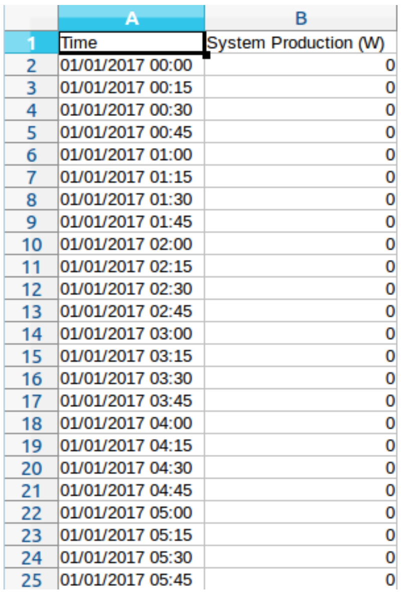


Figure 4: A PV system in IFE

The PV generation data of the PV systems located at Milaveien, Kjeller, Oslo is recorded on the website:https://monitoring.solaredge.com/solaredge-web/p/login.The PV generation data used for researching in this thesis is downloaded from this website. For the purpose to perform real-time monitoring for PV systems, the investigating data in short time interval is required. The data could be obtained on this website are in various form. The PV generation data recorded in 15 min interval is the least time interval available, and some other form report of month, year is available as well.However, the PV generation data recorded in 15 min interval can only be downloaded in one week at a time. For the big data analytics for PV generation and weather conditions, one year period of the data recording is required at least. Thus the PV generation data from 1st January 2017 to 31st December 2017 in 15 minute interval is downloaded. The unit of the pv generation is kwh. Some example of PV generation data sample is as following:

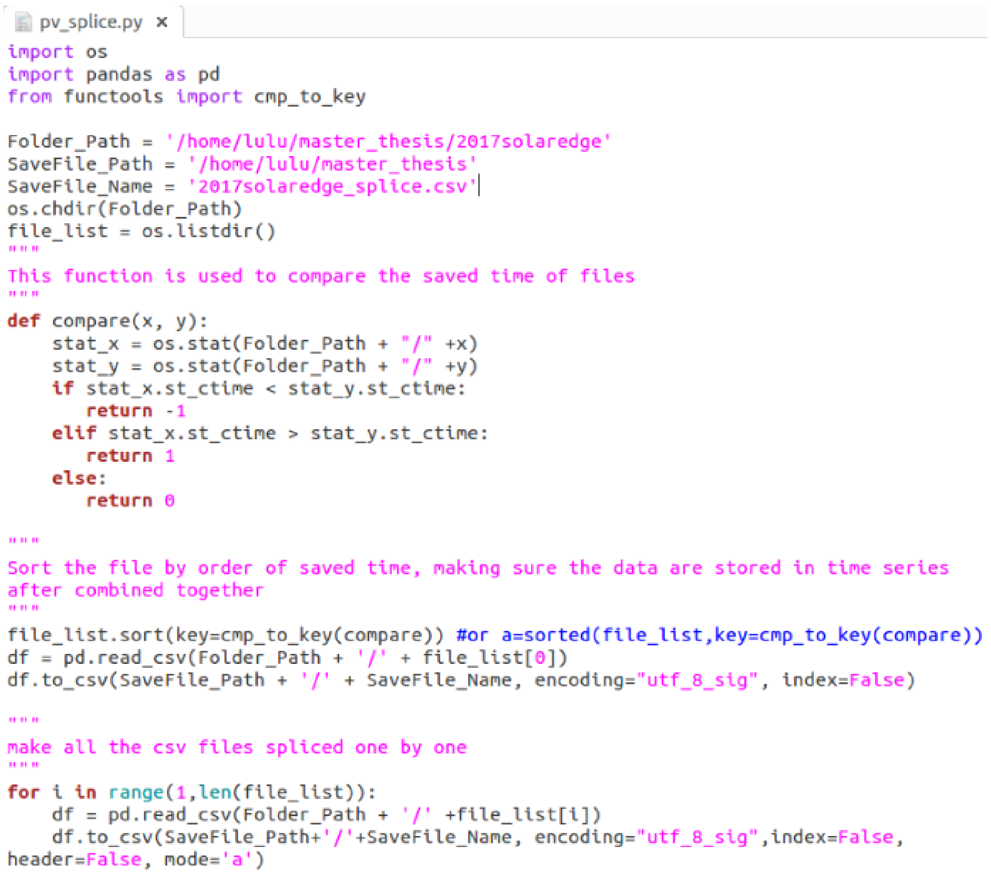


	A	B
1	Time	System Production (W)
2	01/01/2017 00:00	0
3	01/01/2017 00:15	0
4	01/01/2017 00:30	0
5	01/01/2017 00:45	0
6	01/01/2017 01:00	0
7	01/01/2017 01:15	0
8	01/01/2017 01:30	0
9	01/01/2017 01:45	0
10	01/01/2017 02:00	0
11	01/01/2017 02:15	0
12	01/01/2017 02:30	0
13	01/01/2017 02:45	0
14	01/01/2017 03:00	0
15	01/01/2017 03:15	0
16	01/01/2017 03:30	0
17	01/01/2017 03:45	0
18	01/01/2017 04:00	0
19	01/01/2017 04:15	0
20	01/01/2017 04:30	0
21	01/01/2017 04:45	0
22	01/01/2017 05:00	0
23	01/01/2017 05:15	0
24	01/01/2017 05:30	0
25	01/01/2017 05:45	0

Figure 5: Example of PV generation data

As the location of Oslo, Norway is in the north of the world, close to the Arctic circle, the night time is quite long in winter while the day time is quite long in summer. Thus most of the values of PV generation is 0 in winter time. On contrary, not much 0 values of PV generation for summer time.

Since the data of PV generation can only be downloaded from the website in one week at a time. The collected of the PV generation for the year 2017 is separated in 53 files . In order to combine the 53 files, a python program is implemented to splice these files. The code for combining the data is show in the figure below:



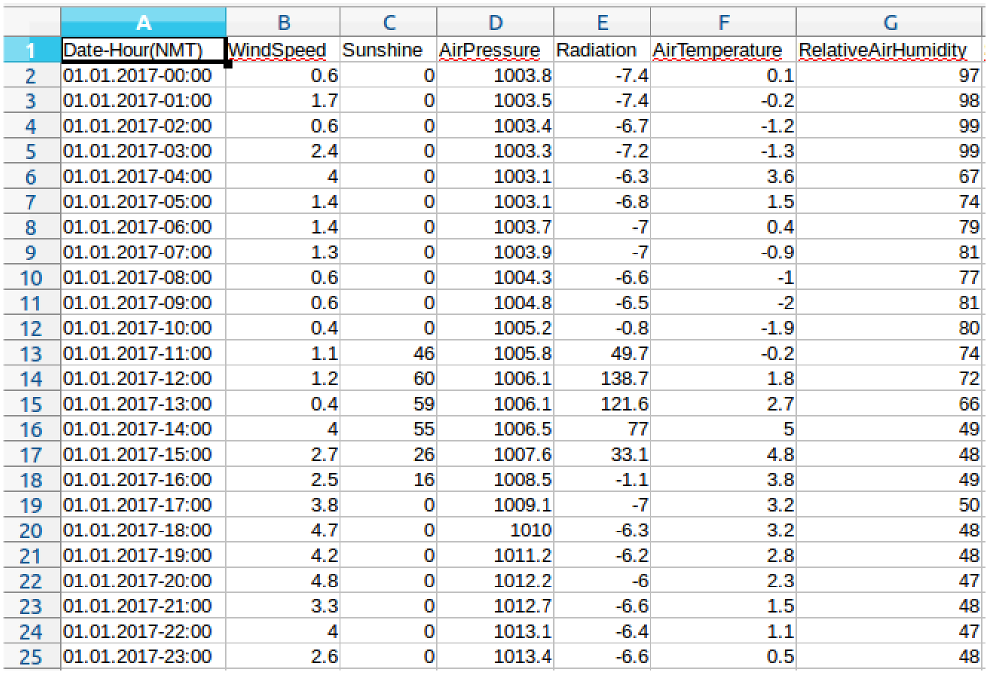
pv_splice.py

```python
import os
import pandas as pd
from functools import cmp_to_key

Folder_Path = '/home/lulu/master_thesis/2017solaredge'
SaveFile_Path = '/home/lulu/master_thesis'
SaveFile_Name = '2017solaredge_splice.csv'
os.chdir(Folder_Path)
file_list = os.listdir()
"""
This function is used to compare the saved time of files
"""
def compare(x, y):
    stat_x = os.stat(Folder_Path + "/" +x)
    stat_y = os.stat(Folder_Path + "/" +y)
    if stat_x.st_ctime < stat_y.st_ctime:
       return -1
    elif stat_x.st_ctime > stat_y.st_ctime:
       return 1
    else:
       return 0

"""
Sort the file by order of saved time, making sure the data are stored in time series
after combined together
"""
file_list.sort(key=cmp_to_key(compare)) #or a=sorted(file_list,key=cmp_to_key(compare))
df = pd.read_csv(Folder_Path + '/' + file_list[0])
df.to_csv(SaveFile_Path + '/' + SaveFile_Name, encoding="utf_8_sig", index=False)

"""
make all the csv files spliced one by one
"""
for i in range(1,len(file_list)):
    df = pd.read_csv(Folder_Path + '/' +file_list[i])
    df.to_csv(SaveFile_Path+'/'+SaveFile_Name, encoding="utf_8_sig",index=False,
header=False, mode='a')
```

Figure 6: Splice separated data files

For investigating the relationship between weather and the PV generation, the weather data of the same year(2017) is required. From the background study, it is acknowledged many weather parameters have influence on the PV power generation, radiation is one of the important factor. However not all the meteorological station observes this parameter nor observation period is not long enough. After searching for the observing stations around IFE, the closest one is the observing station located at Blindern, Oslo. The distance between Blinder, Oslo to Kjeller, Oslo is 25 Kilometers.

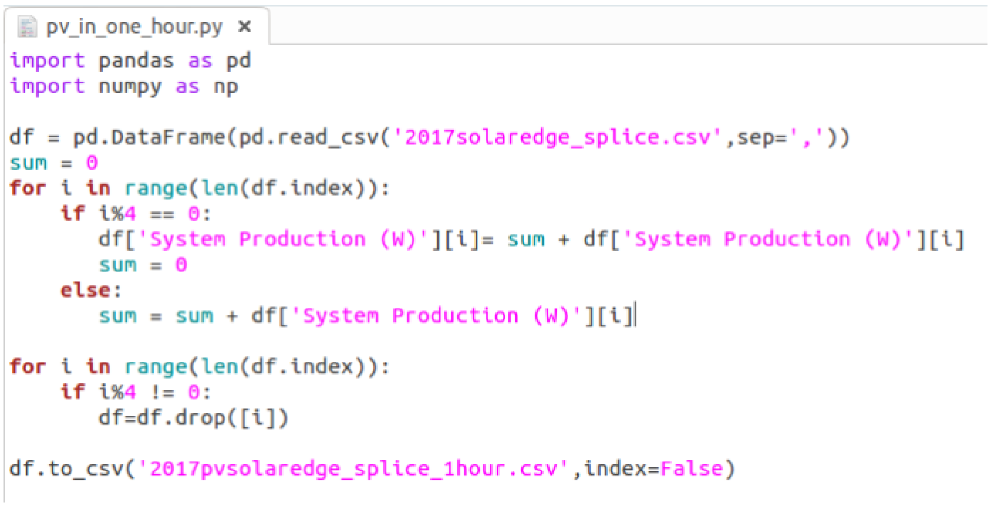
Thus, the history weather data are collected from Blindern observing station. The history weather data recording can be found on the website http://eKlima.no. The measured parameters include air temperature, wind speed measured at 10 meters above ground, air pressure measured at sea level, the minutes of sunshine over the last hour and relative air humidity, in addition, shortwave radiation is also measured at Blindern observation station. Shortwave radiation which is also called global radiation is recorded by mean value over last hour. Shortwave radiation used in this research is measured from above. The example of the weather data sample is show in the following figure :



	A	B	C	D	E	F	G
1	Date-Hour(NMT)	WindSpeed	Sunshine	AirPressure	Radiation	AirTemperature	RelativeAirHumidity
2	01.01.2017-00:00	0.6	0	1003.8	-7.4	0.1	97
3	01.01.2017-01:00	1.7	0	1003.5	-7.4	-0.2	98
4	01.01.2017-02:00	0.6	0	1003.4	-6.7	-1.2	99
5	01.01.2017-03:00	2.4	0	1003.3	-7.2	-1.3	99
6	01.01.2017-04:00	4	0	1003.1	-6.3	3.6	67
7	01.01.2017-05:00	1.4	0	1003.1	-6.8	1.5	74
8	01.01.2017-06:00	1.4	0	1003.7	-7	0.4	79
9	01.01.2017-07:00	1.3	0	1003.9	-7	-0.9	81
10	01.01.2017-08:00	0.6	0	1004.3	-6.6	-1	77
11	01.01.2017-09:00	0.6	0	1004.8	-6.5	-2	81
12	01.01.2017-10:00	0.4	0	1005.2	-0.8	-1.9	80
13	01.01.2017-11:00	1.1	46	1005.8	49.7	-0.2	74
14	01.01.2017-12:00	1.2	60	1006.1	138.7	1.8	72
15	01.01.2017-13:00	0.4	59	1006.1	121.6	2.7	66
16	01.01.2017-14:00	4	55	1006.5	77	5	49
17	01.01.2017-15:00	2.7	26	1007.6	33.1	4.8	48
18	01.01.2017-16:00	2.5	16	1008.5	-1.1	3.8	49
19	01.01.2017-17:00	3.8	0	1009.1	-7	3.2	50
20	01.01.2017-18:00	4.7	0	1010	-6.3	3.2	48
21	01.01.2017-19:00	4.2	0	1011.2	-6.2	2.8	48
22	01.01.2017-20:00	4.8	0	1012.2	-6	2.3	47
23	01.01.2017-21:00	3.3	0	1012.7	-6.6	1.5	48
24	01.01.2017-22:00	4	0	1013.1	-6.4	1.1	47
25	01.01.2017-23:00	2.6	0	1013.4	-6.6	0.5	48

Figure 7: example of weather data

As we can see from the example data, the weather data is in one hour interval, which is the smallest time interval of data recording in http://eKlima.no. In order to make the PV generation data in corresponding with weather data at each time point. An extra processing is required on the PV generation data set. The method adopted in solving this problem is getting the sum of the PV generation in each hour, and remove the record of the other time point except the whole hour point. The code for this processing is as below.

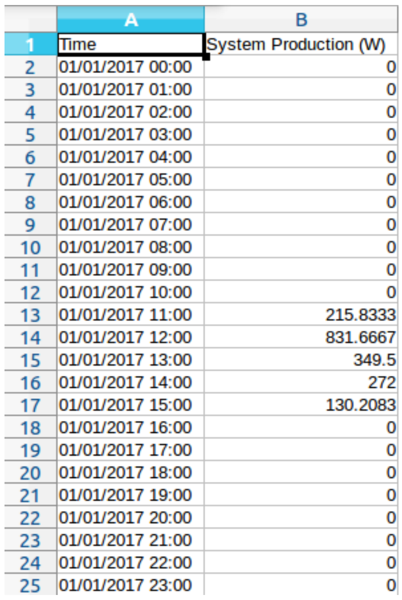


pv_in_one_hour.py ×

```
import pandas as pd
import numpy as np

df = pd.DataFrame(pd.read_csv('2017solaredge_splice.csv',sep=','))
sum = 0
for i in range(len(df.index)):
    if i%4 == 0:
        df['System Production (W)'][i]= sum + df['System Production (W)'][i]
        sum = 0
    else:
        sum = sum + df['System Production (W)'][i]

for i in range(len(df.index)):
    if i%4 != 0:
        df=df.drop([i])

df.to_csv('2017pvsolaredge_splice_1hour.csv',index=False)
```

Figure 8: Code for making the PV data in one hour interval

A new PV generation data set 8760 data samples is produced after the transformation. The PV generation is turned in hour interval as shown in the following figure:



	A	B
1	Time	System Production (W)
2	01/01/2017 00:00	0
3	01/01/2017 01:00	0
4	01/01/2017 02:00	0
5	01/01/2017 03:00	0
6	01/01/2017 04:00	0
7	01/01/2017 05:00	0
8	01/01/2017 06:00	0
9	01/01/2017 07:00	0
10	01/01/2017 08:00	0
11	01/01/2017 09:00	0
12	01/01/2017 10:00	0
13	01/01/2017 11:00	215.8333
14	01/01/2017 12:00	831.6667
15	01/01/2017 13:00	349.5
16	01/01/2017 14:00	272
17	01/01/2017 15:00	130.2083
18	01/01/2017 16:00	0
19	01/01/2017 17:00	0
20	01/01/2017 18:00	0
21	01/01/2017 19:00	0
22	01/01/2017 20:00	0
23	01/01/2017 21:00	0
24	01/01/2017 22:00	0
25	01/01/2017 23:00	0

Figure 9: PV generation data in one hour interval

4.2 data preprocessing

4.2.1 missing values

The data set of history weather data used for research contains some missing values , encoded as blanks. There are two basic strategies in dealing with the missing values. One of the strategy is to use the incomplete data set by discarding the entire rows or/and columns containing missing values.

Obviously, this strategy is simple and fast, but this comes at the price of losing valuable data even though it is not complete. Another strategy is to impute the missing values by calculating from the know part of data in the data set, such as by using the mean value, median value of the row or column in which the missing value is located, or by using the the most frequent value of that row or column.

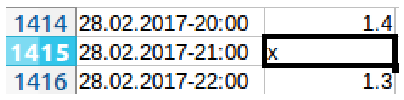
In scikit-learn library, the function *sklearn.preprocessing.Imputer* can perform the completing of missing values in the three strategies:

• If "mean", then replace missing values by using the mean value along the axis.

• If "median", then replace missing values by using the median value along the axis

• If "most_frequent", then replace missing by using the most frequent value along the axis.

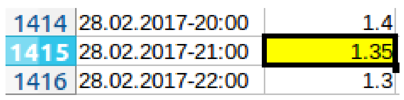
By looking into the characteristic of history weather data set, it is changed slowly with time and recorded with the order of time. Thus, the strategy chosen in this research is to impute the missing values by calculating the mean value of the the last time point recored and the next time point recored of the missing value in time series. An example is as the figure below:



1414	28.02.2017-20:00	1.4
1415	28.02.2017-21:00	x
1416	28.02.2017-22:00	1.3

Figure 10: Example of a missing value

By adopting the strategy filling in the missing value by calculating with the known part, this value will be filled in as the figure below:



1414	28.02.2017-20:00	1.4
1415	28.02.2017-21:00	1.35
1416	28.02.2017-22:00	1.3

Figure 11: An example of dealing with missing value

4.2.2 normalization and standardization

In a training process, if the magnitude of data of features are in big difference, the result might be dominated by one of several features in bigger magnitude instead of all of the features. To solve this problem, the data need to be preprocessed before the training process in order to make the data in a same magnitude. Data normalization and standardization are two general strategy to accomplish this process. Another reason to do normalization and standardization is to improve the speed of convergence of the algorithm.

One of the normalization method is to scale data in linear function to make it fall into a small specific range, which is also called Min-max normalization or 0-1 normalization, and the range is set at [0,1]. The formula of this function is :

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

This method achieve the scaling of the original data, where X_{norm} is the normalized data, X is the original data, and X_{max} and X_{min} are the maximum and minimum values of the original data set respectively. In scikit-learn library, the class $sklearn.preprocessing.MinMaxScaler$ has this method.

Standardization is a method to scale original data by giving a mean value and a standard deviation, Z- score standardization (zero-mean normalization) is one of the standardization method. It set the mean value as 0, and standard deviation as 1. The formula of Z-score standardization is :

$$x^* = \frac{x - \mu}{\sigma}$$

In this formula, x^* is the standardized data, x is the original data, μ is the mean value of the data set, and σ is the standard deviation of the data set. Z- score standardization require the original data set is close to Gaussian distribution. After the processing, the data set meets the standard normal distribution, the mean value is 0, and standard deviation as 1 .

Z- score standardization is used in data preprocessing in this research. Compared with Min-max normalization(0-1 normalization), Z-score standardization has little affect to the later analysis like PCA (principal component analysis) and other covariance analysis or distance analysis. In scikit-learn library, the class $sklearn.preprocessing.StandardScaler$ has this method.

4.3 Split data set

In general, the difference between predicted out put and the real output of the machine learning model is called ‘ error ’, the error of a machine learning model on training set is called ’training error’ or ‘ empirical error ’ , the error on new data sample is called ‘ generalization error ’.

Usually, an experimental test is taken to evaluate a machine learning model and compare the generalization capability among some machine learning models. In order to do this experimental test, a testing set is required. The testing error for the machine learning model on the testing set is called ‘ testing error ’ , which is very comparable to generalization error. Thus, the testing error is often seen as approximation of generalization error.

In practice, the testing set is usually obtained by independently identically distributed sampling from the real data distribution. Thus, splitting the data set into training set and testing set is a step in processing the data. To avoid over-fitting, the data in the training set should not be contained in the testing set.

‘ hold-out ’ is a method to split original data set$D$ into two mutual exclusion set. One of the set is used as training set S, and another is used as testing set T. The relation among this three set is described as: $D = S \cup T$, $S \cap U = \emptyset$.

A machine learning algorithm is trained on the training set S and the testing set T is used to evaluate the testing error which is an estimation of generalization error. To avoid the extra bias caused by the result of splitting data set, it should be noted that try to make the the training set and testing set have the same distribution.

For example, in the classification task, it is basic to keep the percentage of each category similar in training set and testing set. Another question is, there are many ways in splitting data set D. Usually, when using ‘ hold-out ’ method, many times random splitting is adopted, which is performed by repeating the experimental tests and getting the evaluation result by calculating the mean value of the test result of the different splittings.

In addition, the evaluation result we hope to get is the capability of the machine learning model trained on data set D, but in ‘ hold-out ’ method, the data set D is split as training set S and testing set T, this cause a dilemma, if the training set S contains the majority data samples, the machine learning model trained on S is more similar to D, but due to testing set T is too small, the evaluation result is probably not stable and accurate enough; In contrast, if the testing set contains more data sample, the difference between training set S and D is bigger. Thus, this lead to a bigger difference between the evaluated model and the real model trained on D. In balancing this dilemma, a mostly used way is to take 2/3 ~ 4/5 data sample as training data, and the rest used as testing data. The figure below illustrates the 'hold-out' method: In scikit-learn

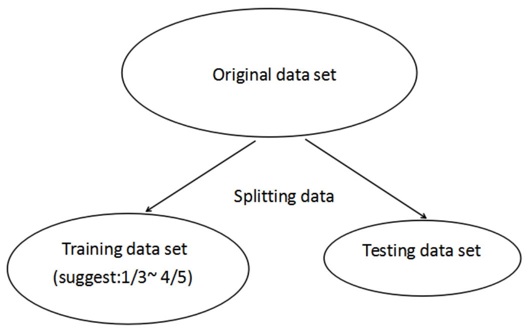


Figure 12: Hold-out method of splitting data set

library, the function $model_selection.train_test_split$ performs this method.

Cross validation is another method in dealing with splitting data. The principle of cross validation is splitting data set D into k exclusive subset of the same size first, that is $D = D_1 \cup D_2 \cup ... \cup D_k, D_i \cap d_j = \emptyset(i \neq j)$, stratified sampling is a suitable method in splitting data set to make each subset D_i has the same distribution. In each experimental test, take k-1 subset combined as training set, and take the rest subset as testing set. In this way, k training set and testing set is obtained, then K times training and testing are performed, in the end the mean value of the k test result is returned. Obviously, the stability and fidelity is dependent on the value of K in a large extent. Thus, it is also called ' k-fold cross validation '. A commonly used value of k is 10, so it is called 10-fold cross validation at this time. Other commonly used value of k is 5, 20 etc. In scikit-learn, $sklearn.model\_selection.Kfold$ is the function that provide train and test set in K-fold. The figure below shows the process of 10-fold cross validation.

Bootstrapping is also a solution in splitting data. It based on bootstrap sampling. For a given data set D containing m data samples, we produce a data set D' by picking up a data from D randomly and copy it into D' , and then putting the data back in D, to make sure it have the chance to be picked up again. Repeating this process m times, we get a data set D' containing m data samples. This is the result of bootstrapping sampling. Obviously, part of the data samples in data set D will be copied in data set D' more than once, while some part of the data in data sample D will not be copied in data set D'. To do an estimation, the probability of one sample which is never be picked up in m sampling

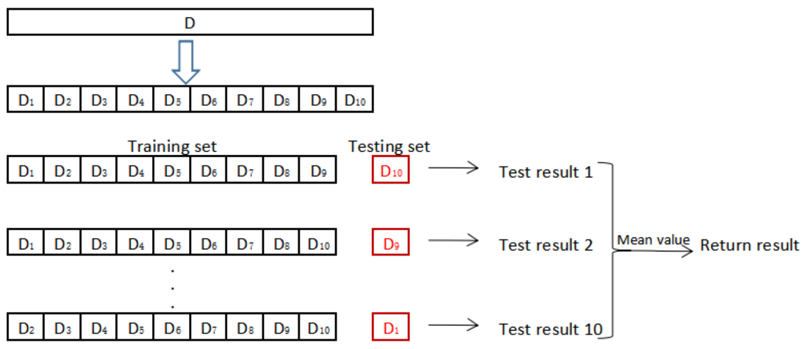


Figure 13: Process of 10-fold cross validation

is $(1 - \frac{1}{m})^m$ take the limit of it is[33] :

$$\lim_{m \to +\infty} (1 - \frac{1}{m})^m \to \frac{1}{e} \approx 0.368$$

By adopting bootstrapping, there are 36.8% data sample in original data set D didn't shown up in data set D' . Then we can take D' as training set and take $D - D'$ as testing set. In this way, the model is being evaluated and the model which is expected to be evaluated both used m training data samples, and there are still nearly 1/3 data samples are not in training set that can be used in testing. This kind of testing result is also called out-of-bag estimate.

Bootstrapping is very useful when the data set is not large, and difficult to split into training set and testing set. However, the distribution of the data set produced by bootstrapping is different with the data distribution of the original data set, this would cause estimation deviation.

4.4 Feature selection

In building a machine learning forecasting model, there are usually a lot of features of the original data, part of them my not be relevant to the learning task, or can be deduced by other features, which is called redundant feature. This kind of features should be removed from the machine leaning model. Otherwise, too much irrelevant and redundant features would cause a rather complicated machine leaning model, and prolong the training time as well. This also lead to a high possibility of over-fitting of the machine learning model. Thus, the performance of machine learning model would be compromised. Therefore, removing the irrelevant and redundant features from a machine learning model is crucial in decreasing the running time and improve the accuracy of the model.

Feature selection is a approach trying to find the subset of the original features. In the process of feature selection, we consider two aspects. One is whether the feature is divergent, if the feature is not divergent, for example, the variance of the feature is close to 0, that means the data samples on this feature has very small difference which is helpless in distinguishing data samples. The other aspect need to be considered is the correlation between the features and the target. It is obvious that the feature which has a high correlation with the target should be selected in a high priority. According the form of the feature selection, we classify the feature selection method into 3 classes. They are Filter, Wrapper and Embedded.

4.4.1 Filter

In Filter type of methods, first give the evaluation scores for each feature by divergence or correlation and then select features by setting threshold of scores or the number of features. The figure shows the process of Filter.

There are some Filter methods which are widely used.

1.Removing features with low variance. For example, the value of one feature is only 0 or 1, and 80% of data samples of this feature is 0, then this feature is not so useful, and if 100% of the data samples of this feature is 0, then we can say this feature is meaningless. This method can only be used when the eigenvalues are discrete variables. If the eigenvalue is continuous variable, the continuous variables need to be discretized before they can be used. But in practice, it is not likely that more than 90% of the data samples have the same value of a feature. Though this method is simple, but it can only accomplish a small part of feature selection task. So his method is usually used as a pre-processing in feature selection, other method is also needed for further feature selection. In scikit-learn library, $sklearn.feature\_selection.VarianceThreshold$ is a feature selector that removes all low variance features.

2.Univariate feature selection. The principle of univariate feature selection is calculating a statistical indicator for each feature separately, and determine which feature is important and then remove the features that are not important. For regression problem, Pearson correlation, distance correlation and mutual information and maximal information coefficient (MIC) could be used.

• Pearson correlation is a simple method to describe the linear relationship between variables. The calculating result is in interval [-1,1], -1 means that the relationship between the variable and the target is

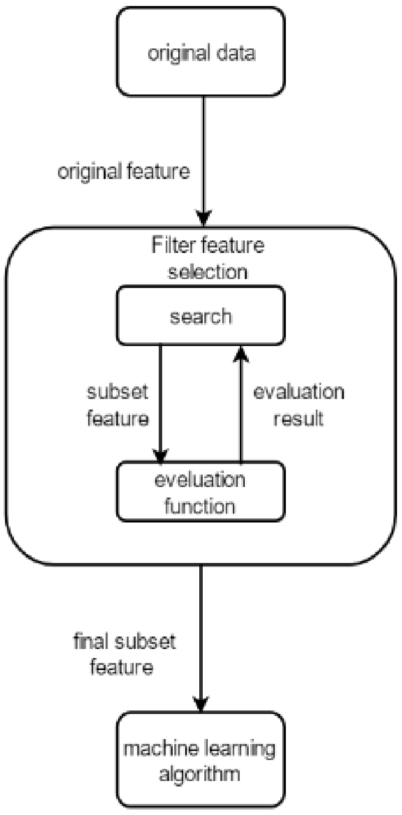


Figure 14: Process of Filter feature selection method

negative correlation completely. +1 means that the relationship between the variable and the target is positive correlation completely. 0 means that there is no only sensitive to linear correlation, even the relationship is one to one correspondence, the result is still close to 0. In scikit-learn library, $sklearn.feature\_selection.f\_regression$ is a function for univariate linear regression test.

• Distance correlation method can overcome the defect of Pearson. Distance correlation measurement methods such as Euclidean distance, Manhattan Distance, Corner cosine, Chebyshev distance and Hamming distance all can be used in measure the distance between variables. For example, to measure the relationship between x and x^2, even the Pearson correlation result is 0, wen can't say they are independent variables, they might be no-linear correlated, but if the distance correlation result is also 0, then it gives enough excuse to say the two variables are independent. Thus, the distance correlation can be seen as a supplement of Pearson correlation.

• Mutual information and maximal information coefficient (MIC) is a efficient method in univariate feature selection. Mutual information is a measurement of the dependency of two random variables, the mutual information of two discrete random variables X and Y could be defined as[34]:

$$I(X;Y) = E[I(X_i;u_i)] = \sum_{x_i \in X} \sum_{y_i \in Y} p(x_i, y_j) log \frac{p(x_i, y_j)}{p(x_i)p(y_j)}$$

where $p(x, y)$ is the joint probability function of X and Y, and $p(x_i)$ and $p(y_i)$ are the marginal probability distribution functions of X and Y respectively. MIC is a method solve problem when the data set is continuous value, it is based on the principle of MI, it does a discretization of the continuous value first and then transforms MI result into a measurement interval [0,1]. If the MIC result is 0, it means the two variables are independent, the bigger the value is, the higher the dependency between the two variables.

In scikit-learn library, $sklearn.feature\_selection.mutal\_info\_regression$ is a function that estimate mutual information for a continuous target variable. Here is the example of feature selection by taking the method $sklearn.feature\_selection.f\_regression$ and method $sklearn.feature\_selection.mutal\_info$ result is shown in the figure below:

From the figure we can conclude that the feature *Radiation* has a best score both on T-test and mutual information, while the feature *Airpressure* has the a worst score both on T-test and mutual information.

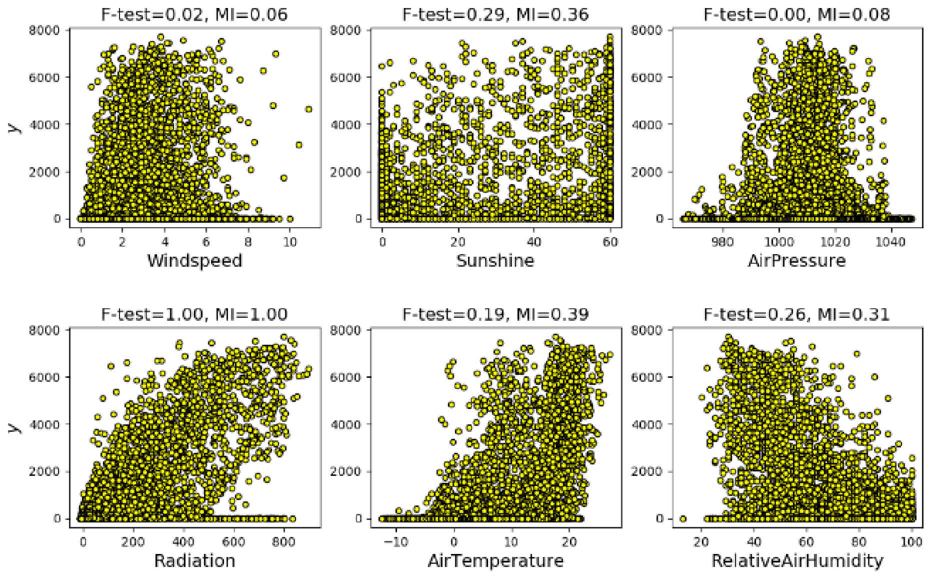


Figure 15: comparison of F-test and mutual information

4.4.2 Wrapper

In Wrapper type methods, feature selection is performed by selecting some features or remove some features according to the evaluation of the machine learning model every time. The process of Wrapper is :

Recursive feature elimination (RFE) is a wrapper type method, RFE uses a base model to train the data set multiple times, then remove several features after each training based on the weights of features. For example, the scikit-learn perform the RFE as: if an external estimator give the weights to features (e.g., the coefficients of a linear model), training the estimator on the initial data set of features and the get the importance of the features through attribute $coef\_$ or $feature\_importance\_$. The least important feature would be removed from the feature set. This procedure is repeated recursively until the desired number of feature is obtained eventually.

In scikit-learn library, $sklearn.feature\_selection.RFE$ performs feature ranking with recursive feature elimination, while the function $sklearn.feature\_selection.R$ also used cross-validated in selecting the best number of features.

By adopting the method $sklearn.feature\_selection.RFE$ in solving the problem of weather feature selection. A ranking of the features is obtained, the code and the result is in the following figures:

From the result of the feature ranking we can get the conclusion that the importance of the features is: $Sunshine > Radiation > windspeed > AirTemperature > relativehumidity > AirPressure$. The least impor-

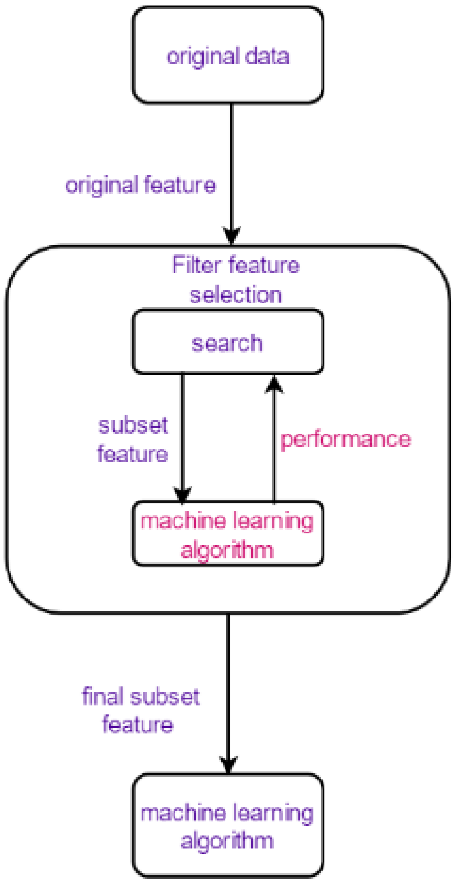
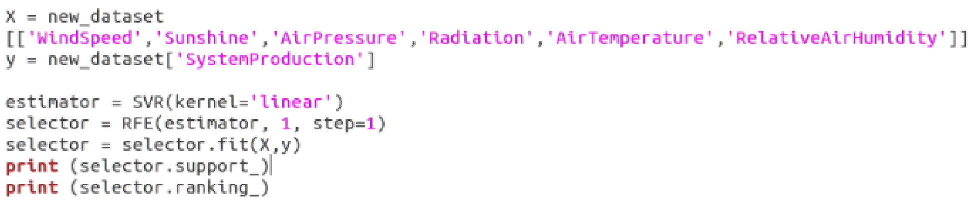
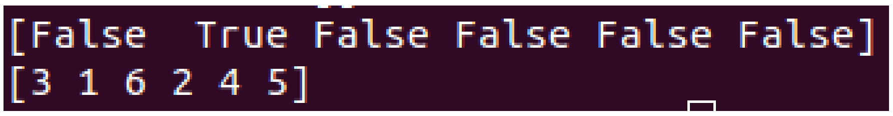


Figure 16: Process of Wrapper feature selection method



```
X = new_dataset
[['WindSpeed','Sunshine','AirPressure','Radiation','AirTemperature','RelativeAirHumidity']]
y = new_dataset['SystemProduction']

estimator = SVR(kernel='linear')
selector = RFE(estimator, 1, step=1)
selector = selector.fit(X,y)
print (selector.support_)
print (selector.ranking_)
```

Figure 17: Code for the features ranking

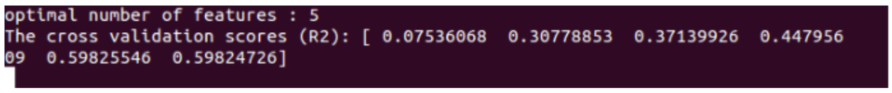


```
[False  True False False False False]
[3 1 6 2 4 5]
```

Figure 18: Result for the feature ranking

tant feature is the $AirPressure$, this result is the same with the result of T-test and mutual information.

In addition, another method $sklearn.feature\_selection.RFECV$ is also used in evaluating the features, cross validation is used in this method, and the evaluation metrics adopted is test is R^2 (The metrics are described in chapter 5). The result of this testing is in the figures below:



```
optimal number of features : 5
The cross validation scores (R2): [ 0.07536068  0.30778853  0.37139926  0.447956
09  0.59825546  0.59824726]
```

Figure 19: Result for the feature scoring

To show this process, a visualization is performed, the number of features is used as x-axis, and the score of the features is taken as y-axis, the visualization result is shown in the figure below:

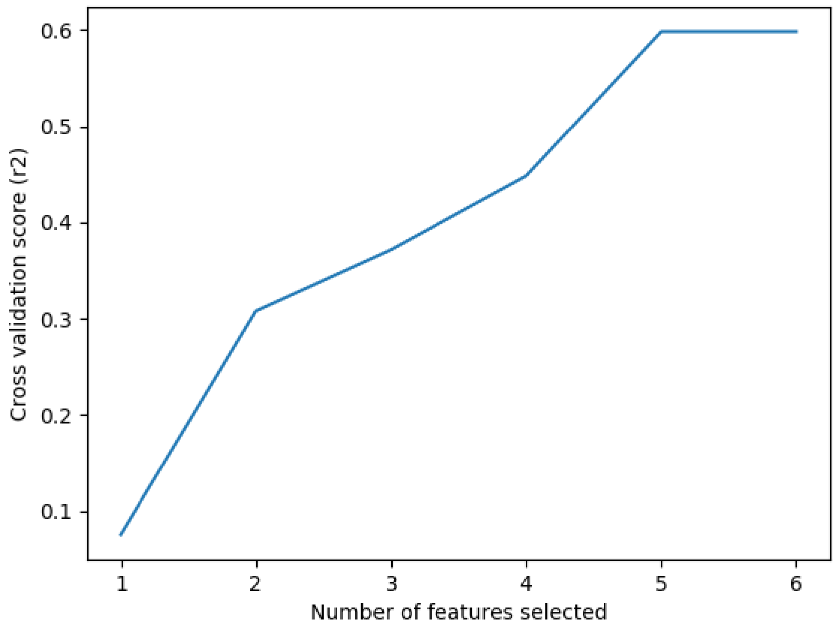


Figure 20: visualization for the feature scoring

In summary of all the test, it indicates that removing the feature $AirPressure$ is a better decision

4.4.3 Embedded

Univariate feature selection measure the correlation between each feature with the variable, another widely used feature selection method is based on the machine learning model. Many machine learning methods can evaluate the features itself, such as L1-based feature selection, randomized sparse model, Tree-based feature selection.

In scikit-learn library, $sklearn.feature\_selection.SelectFromModel$ is a class for selecting features based on important weights. If using the feature selection method, the estimator must have attribute$feature_importances_$ or $coef\_$ after fitting.

4.5 Dimension reduction

Original data set usually contains redundant information and noise information. One way to reduce the redundant information and noise is reducing the data dimension which is called dimension reduction. Dimension reduction can be seen as a process by adopting a mapping method to map high-dimensional data set to low-dimension.

4.5.1 PCA

Principal component analysis(PCA) is a linear technique for dimension reduction. It performs a linear mapping of original data set to a lower-dimension space in such a way that maximizing the variance of the data in lower-dimension representation. In this way, data dimensions are reduced and main characteristics of original data points are kept. The new eigen vectors are generated from part of features which have the biggest variance of the original data. Assume that W is the mapping direction in the target subspace (called mapping vector). It is also unit vector. Maximizing the variance of mapped data. The formula is[33] :

$$max_W \frac{1}{m-1} \sum_{i=1}^{m} ((x_i - \bar{x})^T W)^2$$

In this formula, m is the number of data sample, x_i is vector expression of the data sample i. λ is used to represent:

$$\frac{1}{m-1} \sum_{i=1}^{m} ((x_i - \bar{x})^T W)^2$$

A is used to represent:

$$\frac{1}{m-1} \sum_{i=1}^{m} (x_i - \bar{x})(x_i - \bar{x})^T$$

Thus, above formula would be written as:

$$\lambda = W^T A W$$

For W is unit vector, $W^T W = 1$, two side of the above formula is multiplied by W from left is:

$$W\lambda = \lambda W = W W^T A W = AW$$

That is :

$$AW = \lambda W$$

It can be seen from this formula, λ is the eigenvalues of A, W is eigenvector. The best mapping vector is the eigen vector corresponding with the biggest λ Therefore, the only thing need to be done is finding the solution of the covariance matrix of A. From the result, K biggest eigenvalues are got, and the corresponding eigen vectors of the eigen values are the best k-dimension feature. Moreover, the k-dimension new feature are orthogonal.

In summary, the Process of PCA operating is as following:

1.Compute the mean value for each dimension of original data, and all the data sample minus the mean value in the corresponding dimension.

2.Compute the covariance matrix

3.Compute eigenvalues and eigen vectors of the covariance matrix.

4.Sort eigenvalues from largest to smallest

5.Keep the eigen vectors corresponding with the k biggest eigenvalues

6.Convert the data into a new space constructed by k eigen vectors

In scikit-learn library, *sklearn.decomposition.PCA* is the class that perform lower dimension in PCA principle.

4.5.2 LDA

Linear Discriminant analysis(LDA) is a supervised learning algorithm, it can also used as dimension reduction. For a given data set in n dimensions $x(i)(x_1^{(i)}, x_2(i), ...x_n(i))$, there is a expectation or a class label $y^{(i)}$ corresponding with it. The principle of LDA is: Mapping the data samples with its label together into a lower dimension. In this lower dimension, Trying to make the data samples in the same class closer to each other and data samples belong to different class apart. In scikit-learn library, there is a class *sklearn.dicriminant_analysis.LinearDiscriminantAnalysis* can perform this function.

In order to understand the difference idea between PCA and LDA, take an example, there is a data set in two dimensions x_1 and x_2, when it is mapped into one dimension by using PCA and LDA, the result is shown as figure below, when PCA method is used, it tries to map the data into a dimension that the data have the biggest variance in the direction, while the LDA method tries to map the data into a dimension which can best distinguish classes of the data samples.

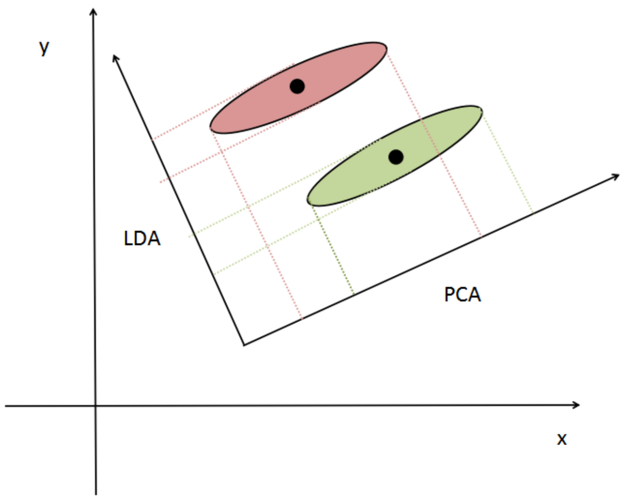


Figure 21: Compare the Principle of PCA and LDA

5 Building Forecasting model

5.1 Linear models

5.1.1 Principle of Linear regression

Linear model a basic machine learning model and has a good comprehensibility. The basic principle of linear model is trying to find a function make prediction by a linear combination of the features. For example, there is n features data sample $\boldsymbol{x} = (x_1; x_2; ...; x_n)$,x_i is the value of x on feature i, then the aim of linear model is trying to find a function:

$$f(x) = w_1x_1 + w_2x_2 + ... + w_nx_n + b \tag{4.1}$$

In a vector form is:

$$F(x) = \boldsymbol{w}^T x + b \tag{4.2}$$

$\boldsymbol{w} = (w_1; w_2; ...; w_n)$.The model will be fixed after *boldsymbolx* and b is learned.

For a given data set D including m data samples, $D = (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_m, y_m)$, for each data sample,$\boldsymbol{x}_i = (x_{i1}; x_{i2}; ...; x_{in})$, $y_i \in R$, then the aim of the linear regression is trying to learn a linear model that predict the output as accurately as possible. This is expressed as:

$$f(x_i) = \boldsymbol{w}^T x_i + b \tag{4.3}$$

trying to make

$$f(x_i) \simeq y_i$$

For estimating $\boldsymbol{w}$ and b, least square method is adopted, the least square method is based on minimizing mean square error which is a frequently used performance measurement. In linear regression, the least square method can also be seen as finding a line which makes a minimal sum of Euclidean distance of all the data sample points. For easy calculation, we use a vector to represent $\boldsymbol{w}$ and b together $\hat{w} = (\boldsymbol{w}; b)$, in corresponding, the data set D is expressed as a $m \cdot (d+1)$ matrix X:

$$X = \begin{pmatrix} x_{11} & x_{12} & ... & x_{1d}1 \\ x_{21} & x_{22} & ... & x_{2d}1 \\ \vdots & \vdots & \ddots & \vdots \\ x_{21} & x_{22} & ... & x_{2d}1 \end{pmatrix} = \begin{pmatrix} x_1^T & 1 \\ x_2^T & 1 \\ \vdots & \vdots \\ x_m^T & 1 \end{pmatrix}$$

Each row in this matrix represents one data sample, the first d elements in each row represent d features of the data set. Then transform the target into a vector form is $\boldsymbol{y} = (y_1; y_2; ...; y_m)$ By adopting the least square method, the expression could be written as[33]:

$$\hat{\boldsymbol{w}}* = argmin_{\hat{w}}(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}})^T(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}}) \tag{4.4}$$

Making $E_{\hat{w}} = (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}})^T(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{w}})$, and then derivative to $\hat{\boldsymbol{w}}$ we get:

$$\frac{\partial E_{\hat{w}}}{\partial \hat{w}} = 2\boldsymbol{X}^T(\boldsymbol{X}\hat{\boldsymbol{w}} - \boldsymbol{y}) \tag{4.5}$$

Making the formula (4.5) is equal to 0,the closed-form of the best answer of $\hat{w}$ will be obtained. Part of the answering process is: if $\boldsymbol{X^T X}$ is a full-rank matrix or positive definite matrix, making the formula (4.5) is equal to 0, then we obtain[33]:

$$\hat{\boldsymbol{w}}* = (\boldsymbol{X^T X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{4.6}$$

In this expression, $(\boldsymbol{X^T X})^{-1}$ is the inverse matrix of$(\boldsymbol{X^T X})$. Taking $\hat{\boldsymbol{x}}_i = (\boldsymbol{x}_i; 1)$. Finally, the expression of linear regression model is[33]:

$$f(\hat{x_i}) = \hat{x_i}^T(\boldsymbol{X^T X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{4.7}$$

However, $\boldsymbol{X^T X}$ is not always a full-rank matrix or positive definite matrix in real life tasks. For example, in some tasks,the number of features is bigger than the number of data samples, this means the number of column is bigger than the number of rows in matrix $\boldsymbol{X}$, obviously, the $\boldsymbol{X^T X}$ is not a full-rank matrix, in this situation, the answer of $\hat{w}$ is more than one, they can all make the mean square error minimized. Which answer to use is depend on inductive preferences. Regularization is one way to deal with this question.

5.1.2 Regularization

Normally, if we have too many features, the machine learning model may fit the training set very well, but fail to generalize on new data samples, this is called over-fitting. There are usually two strategies in solving the problem of over-fitting, one is reducing the number of features, and another is by regularization. The principle of regularization is keeping all the features, but reducing the magnitude of parameters. This works well when there are a lot of features, each feature contributes a bit to the predicting output y. In general, there are two kinds of regularization, L1 regularization and L2 regularization (which is also means penalty). Target function can be seen as adding a regularization to cost function. Cost function is used to describe the difference between real output and the predicted output,write as: $y_i - f(x_i)$. Lasso regression is modeled by using a L1 regularization to linear regression model. The target function of Lasso is [35]:

$$min_w \frac{1}{2n} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_1^2 + \alpha \|\boldsymbol{w}\|_1$$

In this expression, n is the number of data samples, and $\alpha\|\boldsymbol{w}\|_1$ is the penalty item.

While if using L2 regularization to linear regression model, we get the Ridge regression model. The target function of Ridge is[36]:

$$min_w \frac{1}{2n} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2 + \alpha \|\boldsymbol{w}\|_2^2$$

In this expression, n is the number of data samples, and $\alpha\|\boldsymbol{w}\|_2^2$ is the penalty item.

The expression of L1 regularization is : $\alpha \sum_w |w|$ The expression of L2 regularization is : $\alpha \sum_w w^2$

L1 regularization is good at producing a spare model, so that it can help with feature selection, which also lower the possibility of over-fitting. While L2 regularization is good at dealing with over-fitting in another way.

An example is taken to illustrate this, $C = C_0 + L$, C is the target function, C_0 is the original cost function, and $L = \alpha \sum_w |w|$ is the L1 regularization item. So the task now is trying to find the minimum answer of C_0 by the constrain of L. If we consider the situation of two dimensions w_1 and w_2, then $L = |w_1| + |w_2|$. We can draw contours in looking for answers of C_0 in gradient decent method.L1 regularization can also be described on two dimensions.The figure is shown below:

From the figure, we can find that the corner of the L1 square have a bigger chance to touch the C_0, so the weights have a bigger chance to get 0, that is why the L1 regularization is easier to get a spare model, thus that will result in a feature elimination.

Meanwhile, if the L2 regularization is taken, then $L = \alpha \sum_w w^2$, the following figure illustrate the process of taking L2 regularization:

In two dimension, the L2 regularization is a circle. The intersection with C_0 is a lot lower than L1 regularization. The the intersection is usually tend to make weighs as small as possible, smaller weighs often have a better resistance to disturbance, so L2 regularization has the ability to overcome over-fitting in some degree.

5.1.3 Linear model implementation and visualization

The PV generation forecasting is implemented in three linear model respectively, linear regression, lasso, and ridge. In scikit-learn library, the

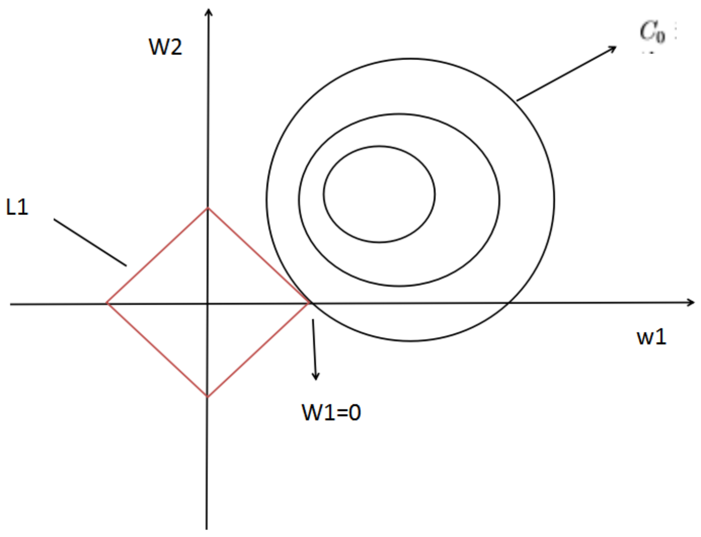


Figure 22: L1 regularization

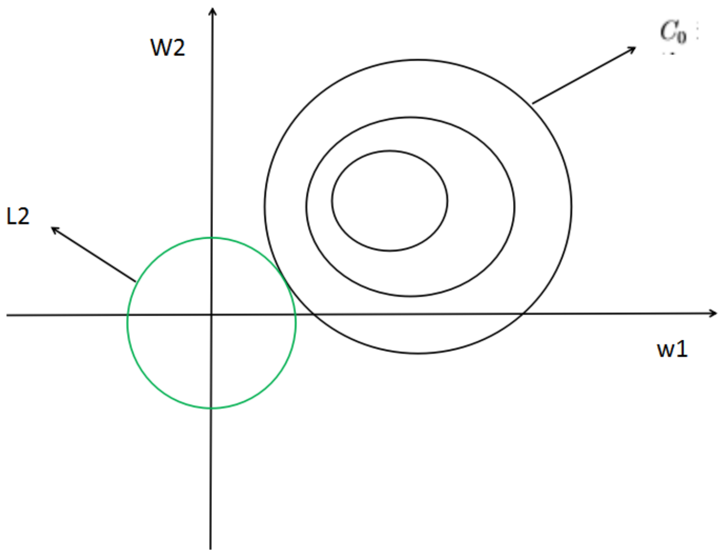
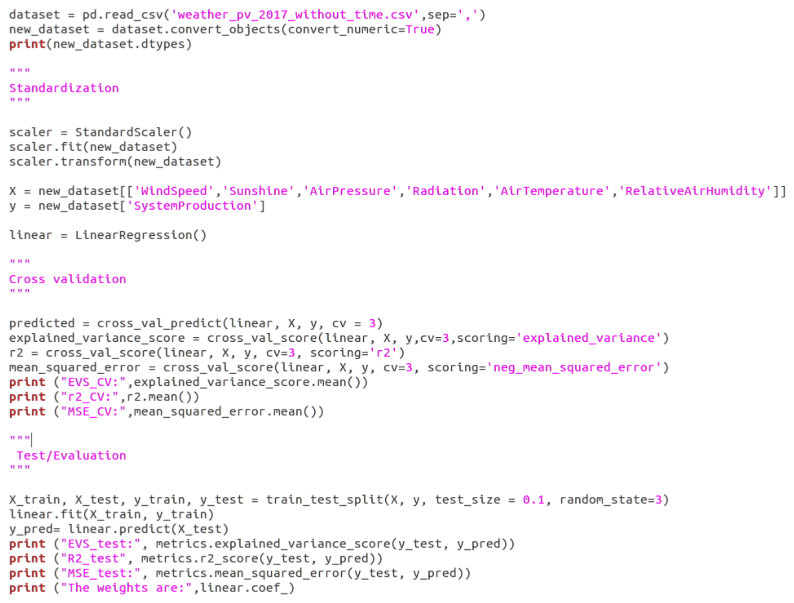


Figure 23: L2 regularization

class *linear_model.LinearRegression* has implemented the linear regression functions.

In splitting the data set, two types of methods are used, one is cross validation, 3 fold cross validation is adopted here and another one is 'hold-out',90% of the data set is used as training data set, and 10% used as testing data set. Part of the code of the linear regression model is as the figure below:

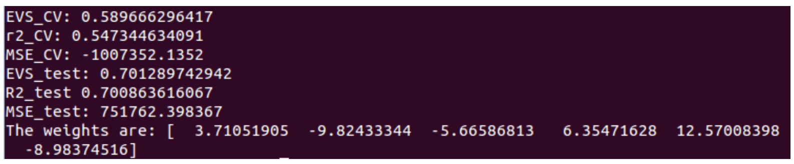


```
dataset = pd.read_csv('weather_pv_2017_without_time.csv',sep=',')
new_dataset = dataset.convert_objects(convert_numeric=True)
print(new_dataset.dtypes)

"""
Standardization
"""

scaler = StandardScaler()
scaler.fit(new_dataset)
scaler.transform(new_dataset)

X = new_dataset[['WindSpeed','Sunshine','AirPressure','Radiation','AirTemperature','RelativeAirHumidity']]
y = new_dataset['SystemProduction']

linear = LinearRegression()

"""
Cross validation
"""

predicted = cross_val_predict(linear, X, y, cv = 3)
explained_variance_score = cross_val_score(linear, X, y,cv=3,scoring='explained_variance')
r2 = cross_val_score(linear, X, y, cv=3, scoring='r2')
mean_squared_error = cross_val_score(linear, X, y, cv=3, scoring='neg_mean_squared_error')
print ("EVS_CV:",explained_variance_score.mean())
print ("r2_CV:",r2.mean())
print ("MSE_CV:",mean_squared_error.mean())

"""
 Test/Evaluation
"""

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state=3)
linear.fit(X_train, y_train)
y_pred= linear.predict(X_test)
print ("EVS_test:", metrics.explained_variance_score(y_test, y_pred))
print ("R2_test", metrics.r2_score(y_test, y_pred))
print ("MSE_test:", metrics.mean_squared_error(y_test, y_pred))
print ("The weights are:",linear.coef_)
```

Figure 24: Part of the code for linear model

There are three measurement are used: MVS(Explained variance Score), R2 score (R^2, coefficient of determination), MSE(Mean Square Error)

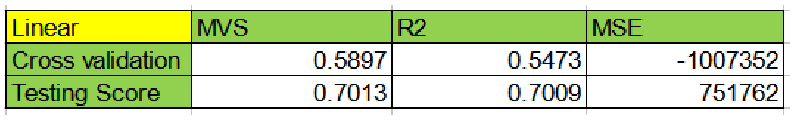
The output of the code are the scores of the three kinds of measurement for cross validation and testing respectively, and the *weights* of the 6 features. The result is as following:



```
EVS_CV: 0.589666296417
r2_CV: 0.547344634091
MSE_CV: -1007352.1352
EVS_test: 0.701289742942
R2_test 0.700863616067
MSE_test: 751762.398367
The weights are: [  3.71051905  -9.82433344  -5.66586813   6.35471628  12.57008398
  -8.98374516]
```

Figure 25: output of the linear model

To make it clear, the scores are summarized in a table as bellow: By taking the measured value as X-axis, and the corresponding predicted



Linear	MVS	R2	MSE
Cross validation	0.5897	0.5473	-1007352
Testing Score	0.7013	0.7009	751762

Figure 26: Scores for linear model

value as y-axis, two figures are plotted for visualization: The cross validation prediction result is shown in the figure bellow: The testing result

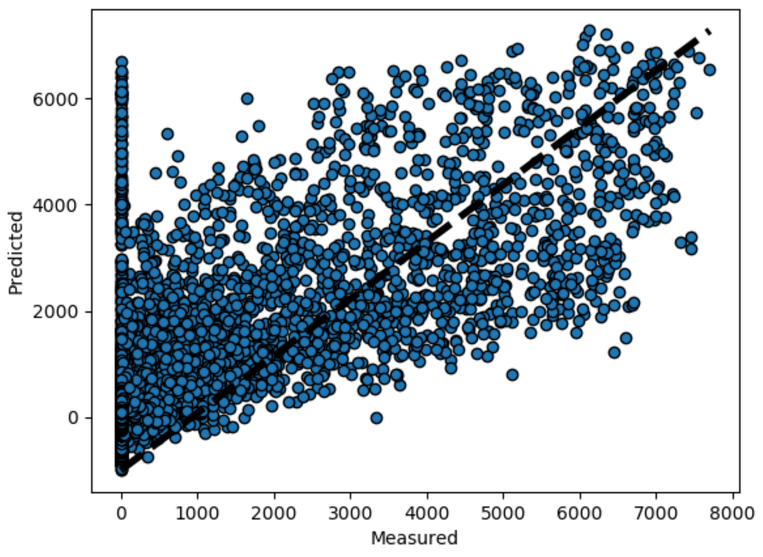


Figure 27: Visualization of the cross validation result of linear model

is shown in the next figure:

5.1.4 Lasso model implementation and visualization

Lasso regression model is based on Linear regression model with L1 regularization. Class *linear_model.Lasso* in scikit-learn provide the implementation of lasso regression. There is a parameter *alpha*, representing the strength of the penalty.The default value of this parameter is 1.0. In this case, the parameter of *alpha* is also set to be 1.0. The code of building the lasso regression model is shown in the figure below, the cross validation and 'hold-out' methods in splitting data set are used. To make sure the training data set and testing data set are the same for different learning models, a $random_state$ is set to be the same in three learning

The same measurements are used as linear model, MVS(Explained variance Score), R2 score (R^2, coefficient of determination), MSE(Mean

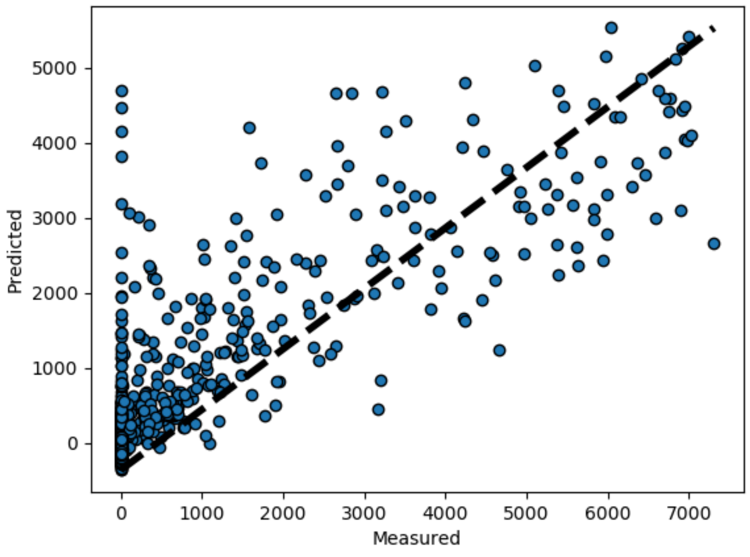
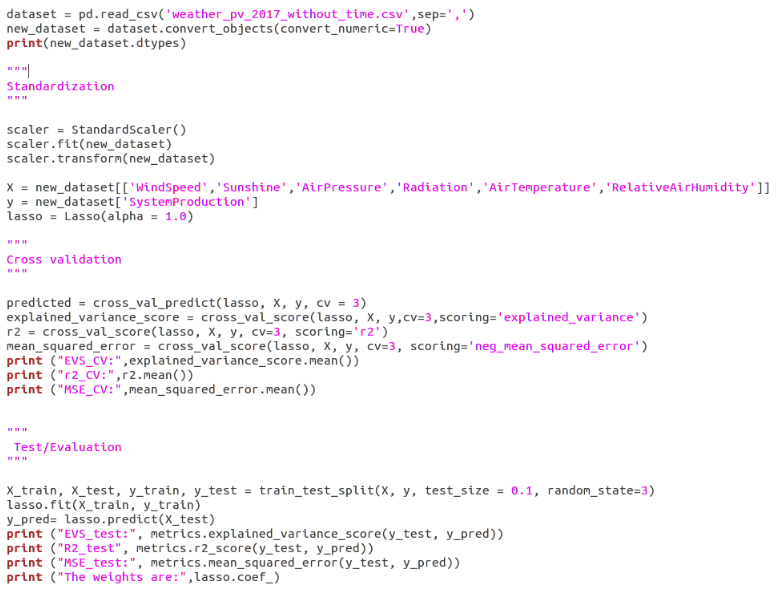


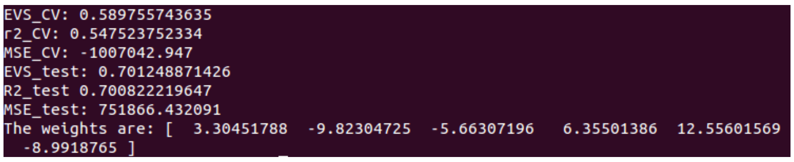
Figure 28: Visualization of the testing result of linear model



```
dataset = pd.read_csv('weather_pv_2017_without_time.csv',sep=',')
new_dataset = dataset.convert_objects(convert_numeric=True)
print(new_dataset.dtypes)

"""
Standardization
"""

scaler = StandardScaler()
scaler.fit(new_dataset)
scaler.transform(new_dataset)

X = new_dataset[['WindSpeed','Sunshine','AirPressure','Radiation','AirTemperature','RelativeAirHumidity']]
y = new_dataset['SystemProduction']
lasso = Lasso(alpha = 1.0)

"""
Cross validation
"""

predicted = cross_val_predict(lasso, X, y, cv = 3)
explained_variance_score = cross_val_score(lasso, X, y,cv=3,scoring='explained_variance')
r2 = cross_val_score(lasso, X, y, cv=3, scoring='r2')
mean_squared_error = cross_val_score(lasso, X, y, cv=3, scoring='neg_mean_squared_error')
print ("EVS_CV:",explained_variance_score.mean())
print ("r2_CV:",r2.mean())
print ("MSE_CV:",mean_squared_error.mean())


"""
 Test/Evaluation
"""

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state=3)
lasso.fit(X_train, y_train)
y_pred= lasso.predict(X_test)
print ("EVS_test:", metrics.explained_variance_score(y_test, y_pred))
print ("R2_test", metrics.r2_score(y_test, y_pred))
print ("MSE_test:", metrics.mean_squared_error(y_test, y_pred))
print ("The weights are:",lasso.coef_)
```

Figure 29: Part of the code for lasso model

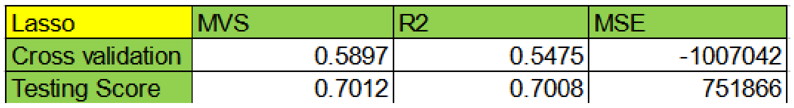
Square Error) for cross validation and testing respectively, and the *weights* of the 6 features. The result is as following:



```
EVS_CV: 0.589755743635
r2_CV: 0.547523752334
MSE_CV: -1007042.947
EVS_test: 0.701248871426
R2_test 0.700822219647
MSE_test: 751866.432091
The weights are: [  3.30451788  -9.82304725  -5.66307196   6.35501386  12.55601569
  -8.9918765 ]
```

Figure 30: output of the lasso model

Summarizing the output in a table as bellow:



Lasso	MVS	R2	MSE
Cross validation	0.5897	0.5475	-1007042
Testing Score	0.7012	0.7008	751866

Figure 31: Scores for lasso model

From the evaluation result, we notice that the performance for the linear model and lasso are only slightly different.

The visualization of cross validation prediction result of lasso regression model is shown in the figure bellow:

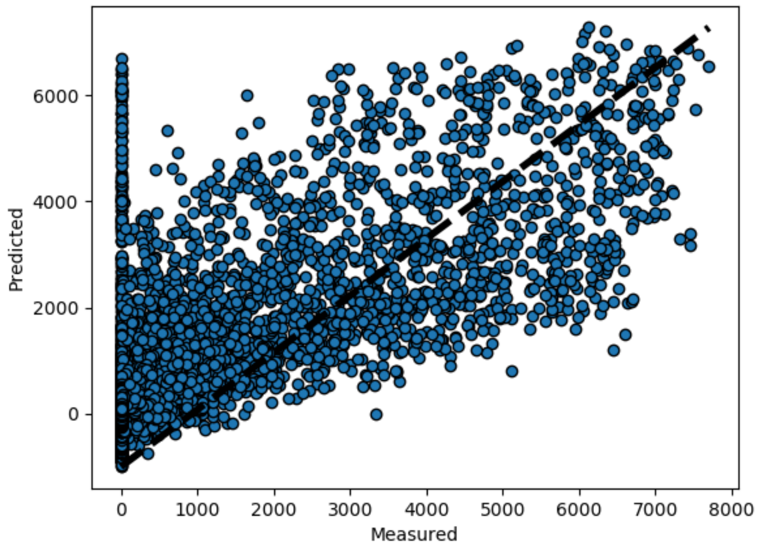


Figure 32: Visualization of the cross validation result of lasso model

The visualization result for testing result of lasso regression model is shown in the next figure:

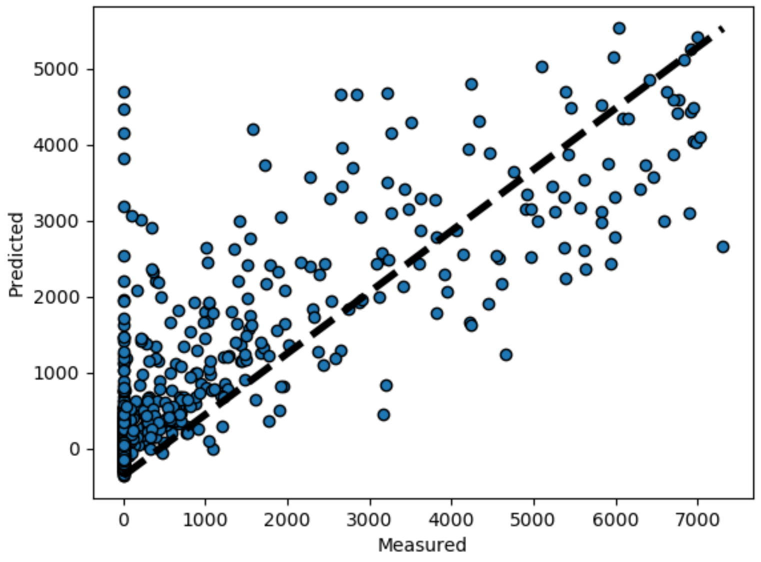


Figure 33: Visualization of the testing result of lasso model

5.1.5 Ridge model implementation and visualization

linear_model.Ridge Ridge regression model is based on linear regression model with L2 regularization. The same with Lasso regression, There is a parameter *alpha*, controlling the strength of the penalty, it is chosen to be 1.0 in this case.The methods of splitting data set are all the same with linear regression model and lasso regression model. The code of building the lasso regression model is shown in the figure below:

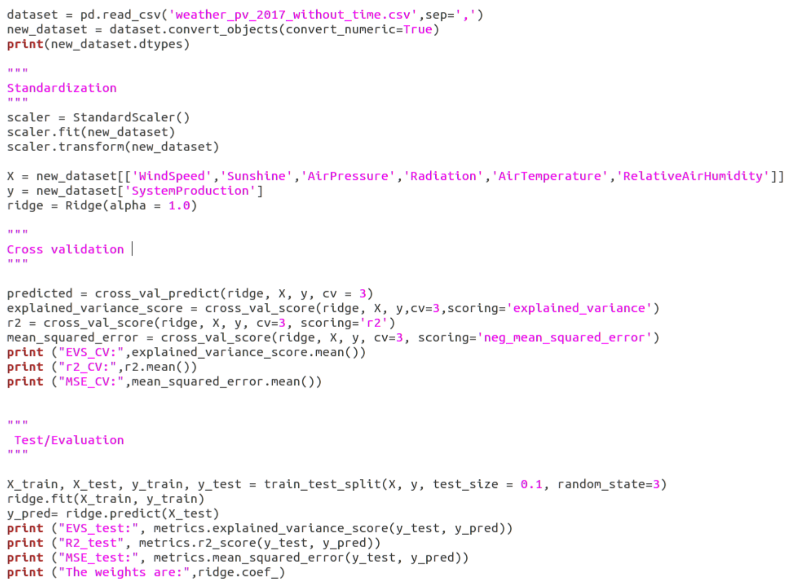
The measurement adopted for evaluating the model performance are the same as linear regression model and lasso regression model, the outputs are MVS(Explained variance Score), R2 score (R^2, coefficient of determination), MSE(Mean Square Error) for cross validation and testing respectively, and the *weights* of the 6 features. The result is as following:

The output is summarized in a table as bellow:

From the result of the measurement scores we found that the performance of ridge regression model is almost the same with linear regression. Except that, when we looked into the *weights* obtained after learning by the three models, they are almost all the same result.

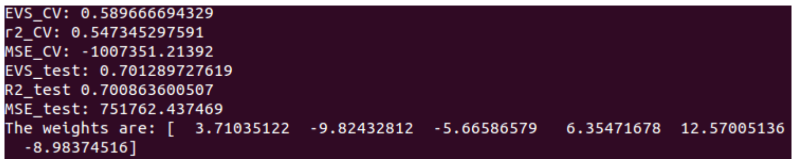
The visualization of cross validation prediction result of ridge regression model is shown in the figure bellow:

The visualization result for testing result of ridge regression model is shown in the next figure:



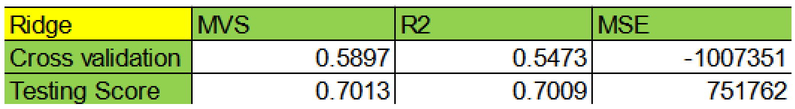
```
dataset = pd.read_csv('weather_pv_2017_without_time.csv',sep=',')
new_dataset = dataset.convert_objects(convert_numeric=True)
print(new_dataset.dtypes)

"""
Standardization
"""
scaler = StandardScaler()
scaler.fit(new_dataset)
scaler.transform(new_dataset)

X = new_dataset[['WindSpeed','Sunshine','AirPressure','Radiation','AirTemperature','RelativeAirHumidity']]
y = new_dataset['SystemProduction']
ridge = Ridge(alpha = 1.0)

"""
Cross validation
"""

predicted = cross_val_predict(ridge, X, y, cv = 3)
explained_variance_score = cross_val_score(ridge, X, y,cv=3,scoring='explained_variance')
r2 = cross_val_score(ridge, X, y, cv=3, scoring='r2')
mean_squared_error = cross_val_score(ridge, X, y, cv=3, scoring='neg_mean_squared_error')
print ("EVS_CV:",explained_variance_score.mean())
print ("r2_CV:",r2.mean())
print ("MSE_CV:",mean_squared_error.mean())


"""
 Test/Evaluation
"""

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state=3)
ridge.fit(X_train, y_train)
y_pred= ridge.predict(X_test)
print ("EVS_test:", metrics.explained_variance_score(y_test, y_pred))
print ("R2_test", metrics.r2_score(y_test, y_pred))
print ("MSE_test:", metrics.mean_squared_error(y_test, y_pred))
print ("The weights are:",ridge.coef_)
```

Figure 34: Part of the code for ridge model



```
EVS_CV: 0.589666694329
r2_CV: 0.547345297591
MSE_CV: -1007351.21392
EVS_test: 0.701289727619
R2_test 0.700863600507
MSE_test: 751762.437469
The weights are: [  3.71035122  -9.82432812  -5.66586579   6.35471678  12.57005136
  -8.98374516]
```

Figure 35: output of the ridge model



Ridge	MVS	R2	MSE
Cross validation	0.5897	0.5473	-1007351
Testing Score	0.7013	0.7009	751762

Figure 36: Scores for ridge model

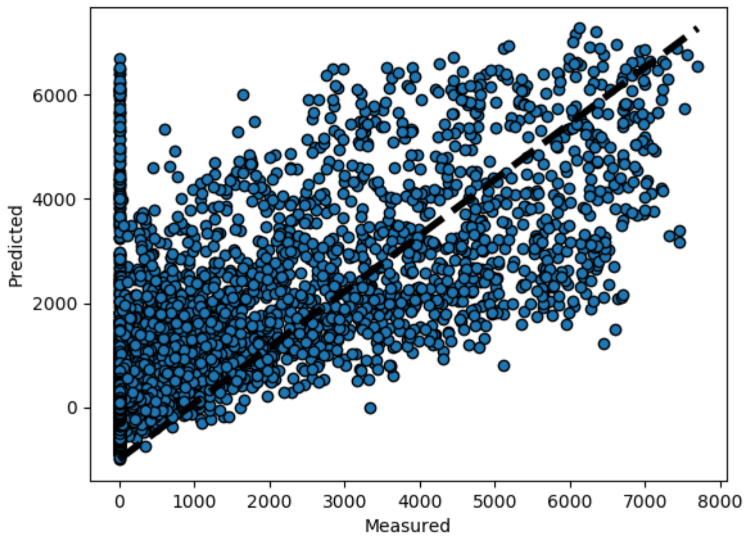


Figure 37: Visualization of the cross validation result of ridge model

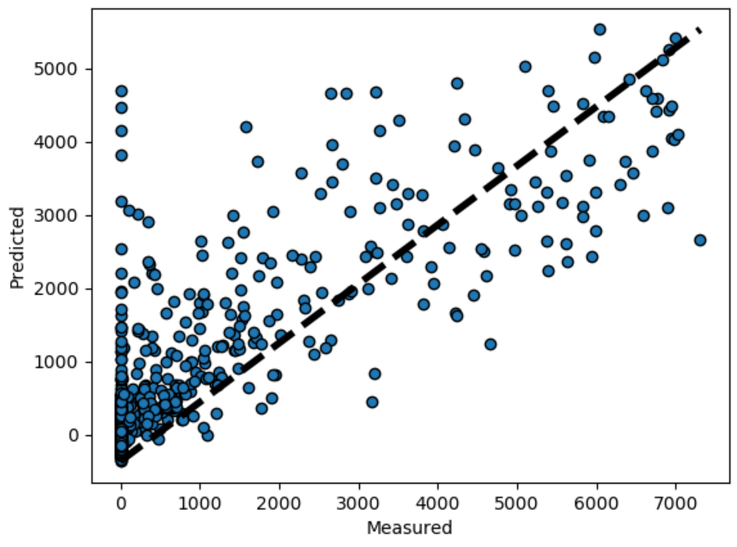


Figure 38: Visualization of the testing result of ridge model

5.2 Support Vector Regression (SVR) model

5.2.1 Principle of SVR

SVR (Support Vector Regression) is a method to solve regression problem which is based on SVM (Support Vector Machine). For a given training data sample $D = (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{X}_m, y_m), y_i \in R$, We hope to get a regression model like $f(\boldsymbol{x}) = \boldsymbol{\omega}^T x + b$, the aim is to make $f(\boldsymbol{x})$ closer with y, $\boldsymbol{\omega}$ and b are parameters need to be fixed.

For the data sample$(\boldsymbol{x}, y)$, in traditional regression algorithms, loss is calculated by the difference between the output $f(\boldsymbol{x})$ of the forecasting model and the real output y. Only when they are equal, the loss is 0. The situation is different in SVR, it allows a bias between $f(\boldsymbol{x})$ and y at most ϵ, loss will be calculated only when the difference between $f(\boldsymbol{x})$ and y is bigger than the bias ϵ. This method is described as the figure:

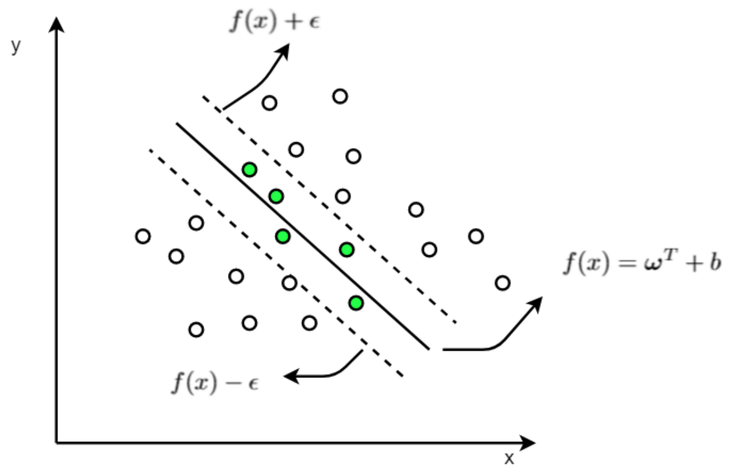


Figure 39: SVR principle, data sample falls into the margin between the two dashed lines doesn't count loss

It means that a width of 2ϵ soft margin is set up in the center of $f(\boldsymbol{x})$, if the predicted result of the training data sample falls into this margin, it is recognized to be correctly predicted. [33]

Thus,the SVR problem could be expressed as [33]:

$$min_{\boldsymbol{\omega},b}\frac{1}{2}\left\|\boldsymbol{\omega}\right\|^2 + C\sum_{i=1}^{m}\iota_\epsilon(f(x_i) - y_i) \tag{4.1}$$

In this expression, C is is the constant value of regularization, ι_ϵ is a ϵ-insensitive loss like shown in the figure below.

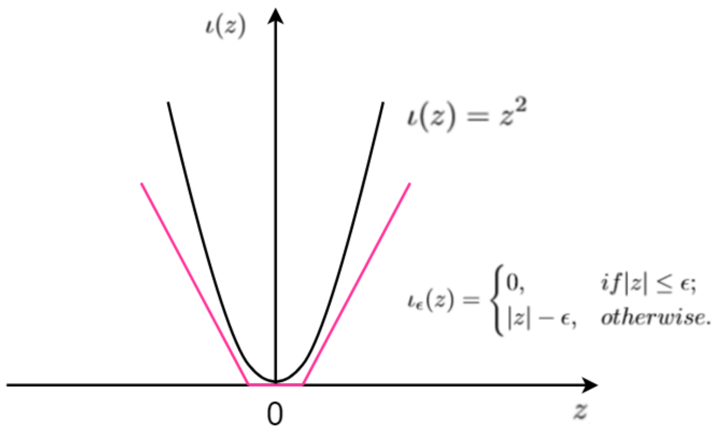


Figure 40: ϵ-insensitive loss function

$$\iota_\epsilon(z) = \begin{cases} 0, & if|z| \leq \epsilon; \\ |z| - \epsilon, & otherwise. \end{cases} \tag{4.2}$$

Bring in two slack variablesξ_i and $\hat{\xi}_i$ to expression (4.1), the expression would be written as [33]

$$min_(\boldsymbol{\omega}, b, \xi_i, \hat{\xi}_i)\frac{1}{2}\left\|\boldsymbol{\omega}\right\|^2 + C\sum_{i=1}^{m}(\xi + \hat{\xi}_i) \tag{4.3}$$

s.t.

$$f(x_i) - y_i \leq \epsilon + \xi_i$$
$$y_i - f(x_i) \leq \epsilon + \hat{\xi}_i$$
$$\xi_i \geq 0, \hat{\xi}_i \geq 0, i = 1, 2, ..., m.$$

The dual problem is obtained by introducing Lagrange multiplier method,and after a series of complex transformation, the SVR problem can be expressed as[33]:

$$f(x) = \sum_{i=1}^{m}(\hat{\alpha}_i - \alpha_i)\kappa(x, x_i) + b \tag{4.4}$$

In this expression, $\kappa(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ is a kernel function. Here training vectors are implicitly mapped into a higher dimensional space by the function ϕ

5.2.2 SVR functions

From the principle of SVR, we can find that the predicting model built by using Support Vector Regression depends only on a subset of the training data, because the cost function for producing the predicting model ignores any training data close to the model prediction.

There are three different implementation methods of Support Vector Regression in scikit learn. linearSVR, SVR and NuSVR.

LinearSVR implementation is faster compared with SVR and NuSVR, but it only uses linear kernel. NuSVR(Nu Support Vector Regression) uses a parameter nu to control the number of support vectors.[37]

In Scikit-learn, four types of kernel functions are provided, they are linear, polynomial, RBF(Radial Basis Function) and sigmoid. Different kernels are specified by keyword kernel at initialization.

•The expression of linear kernel is: $K(\boldsymbol{x}, z) = \boldsymbol{x} \cdot z$, linearSVR can only use this kernel.

•Polynomial kernel is one of the usually used non-linear kernels. The expression is: $K(\boldsymbol{x}, z) = (\gamma \boldsymbol{x} \cdot z + \boldsymbol{r})^d$, in this expression, $\gamma, \boldsymbol{r}, d$ need to be defined and adjusted by users manually when this kernel is used .

• Gaussian kernel is also called Radial Basis Function (RBF), it is the default kernel function in scikit-learn.The expression of RBF is $K(\boldsymbol{x}, z) = exp(-\gamma \left\| \boldsymbol{x} - z \right\|^2)$, in this expression, γ is bigger than 0, it need to be adjusted by users manually when this kernel is used.

• sigmoid kernel is also one of the often used non-linear kernel.The expression of this kernel is: $K(\boldsymbol{x}, z) = tanh(\gamma \boldsymbol{x} \cdot z + \boldsymbol{r})$, parameter γ and $\boldsymbol{r}$ need to be adjusted by user manually in this expression.

5.2.3 Parameter selection

Parameter selection is a crucial step in setting up a machine learning model, it has a big influence on the performance of the estimator, if the parameter is not adjusted good, the performance of the model could be arbitrary worse.

When training a SVR model with Radial Basis Function(RBF) kernel, two parameters should be considered,they are C and gamma.The parameter C is used to trade off misclassification of training samples against simplicity of the decision surface. When the C is low, the decision surface is more smooth, and if the C is high, it aims at making all training samples classified correctly. As for parameter gamma, it defines how much influence a single training sample has to others. A larger gamma makes the other samples are affected closer. Proper choice of C and gamma is very important for the performance of SVR when RBF kernel is used.

If polynomial kernel is used in training a machine learning model, parameter $degree$ and $coef0$ should be carefully adjusted. Because in the expression of polynomial kernel$K(\boldsymbol{x}, z) = (\gamma\boldsymbol{x} \cdot z + \boldsymbol{r})^d$, d is specified by keyword $degree$ and $coef0$ specifies r.

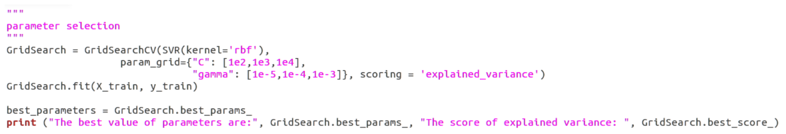
For sigmoid kernel, when it is used in a SVR model,parameter $coef0$ should be considered, it specifies r in the sigmoid expression:$K(\boldsymbol{x}, z) = tanh(\gamma\boldsymbol{x} \cdot z + \boldsymbol{r})$

As for the linear kernel, it is only used in linearSVR, parameter C should be adjusted properly, it controls the penalty of the error term.

Except that, when NuSVR method is used, parameternu should be set properly,it represent an upper bound on the fraction of training errors and lower bound of the fraction of the support vector. It should be in the interval $(0, 1]$. By default 0.5 will be taken.

5.2.4 SVR model implementation

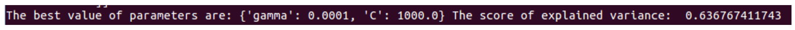
RBF kernel is adopted in the SVR model for prediction the PV generation, there are two parameters need to be fixed for FBF kernel, C and $gamma$, these two parameters must be fit well at the same time in order to get better performance of the learning model. the class *sklearn.model_selection.GridSearchCV* provide the function for exhaustive search over specified parameter values for an estimator. The parameter cv is used for setting the number of folds of cross validation, default value is 3, there is also a measurement should be given by setting the parameter$scoring$. In this case, the parameter default value $cv = 3$ is used, and explained_variance is chosen as the measurement.The code for the parameter selection is:



```
"""
parameter selection
"""
GridSearch = GridSearchCV(SVR(kernel='rbf'),
                   param_grid={"C": [1e2,1e3,1e4],
                               "gamma": [1e-5,1e-4,1e-3]}, scoring = 'explained_variance')
GridSearch.fit(X_train, y_train)

best_parameters = GridSearch.best_params_
print ("The best value of parameters are:", GridSearch.best_params_, "The score of explained variance: ", GridSearch.best_score_)
```

Figure 41: Code for parameter selection

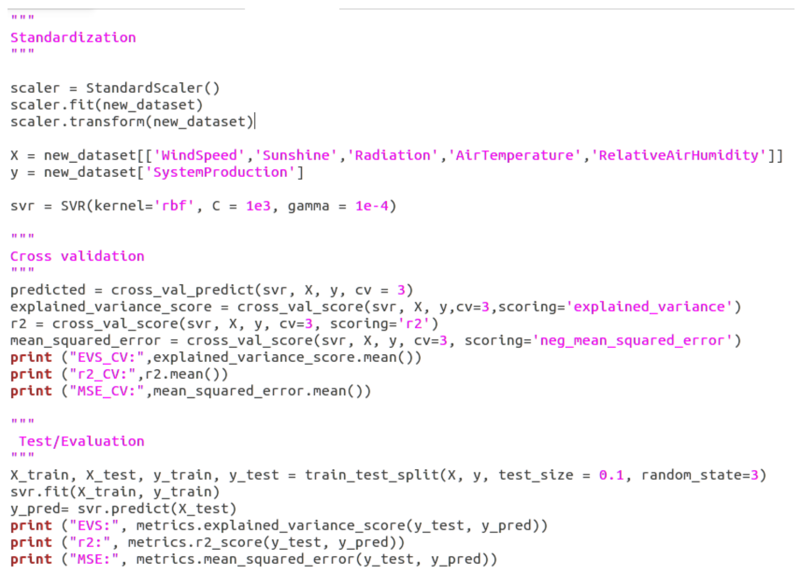
The result of the parameter selection is as the figure below:



```
The best value of parameters are: {'gamma': 0.0001, 'C': 1000.0} The score of explained variance:  0.636767411743
```

Figure 42: Result for parameter selection

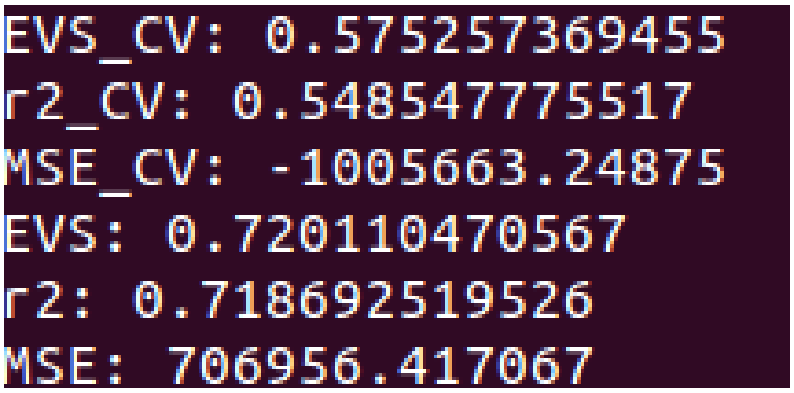
The best value for C is 1000.0, and $gamma$ is 0.0001. Based on the best parameter selected, the model is trained by using the specific values of the parameters c and $gamma$. The data set is split in the same way as linear model. Two kinds of dataset splitting methods are used in testing the learning model, first one is cross validation, 3-fold cross validation is used in this case, and another splitting method is ’hold-out’, 90% of the original data is taken for training data set and the rest 10% data is used as testing data set. For the purpose of doing evaluation of the models in the next chapter, the same random splitting state is used. The measurements adopted are the same as linear model, MVS(Explained variance Score), R2 score (R^2, coefficient of determination), MSE(Mean Square Error). Part of the code of the implementation is as below:



```
"""
Standardization
"""

scaler = StandardScaler()
scaler.fit(new_dataset)
scaler.transform(new_dataset)

X = new_dataset[['WindSpeed','Sunshine','Radiation','AirTemperature','RelativeAirHumidity']]
y = new_dataset['SystemProduction']

svr = SVR(kernel='rbf', C = 1e3, gamma = 1e-4)

"""
Cross validation
"""
predicted = cross_val_predict(svr, X, y, cv = 3)
explained_variance_score = cross_val_score(svr, X, y,cv=3,scoring='explained_variance')
r2 = cross_val_score(svr, X, y, cv=3, scoring='r2')
mean_squared_error = cross_val_score(svr, X, y, cv=3, scoring='neg_mean_squared_error')
print ("EVS_CV:",explained_variance_score.mean())
print ("r2_CV:",r2.mean())
print ("MSE_CV:",mean_squared_error.mean())

"""
 Test/Evaluation
"""
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state=3)
svr.fit(X_train, y_train)
y_pred= svr.predict(X_test)
print ("EVS:", metrics.explained_variance_score(y_test, y_pred))
print ("r2:", metrics.r2_score(y_test, y_pred))
print ("MSE:", metrics.mean_squared_error(y_test, y_pred))
```

Figure 43: Partial Code for implementation a SVR model

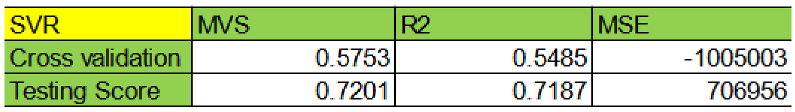
The output shows the validation scores and testing scores of the three different measurement of the SVR learning model :



```
EVS_CV: 0.575257369455
r2_CV: 0.548547775517
MSE_CV: -1005663.24875
EVS: 0.720110470567
r2: 0.718692519526
MSE: 706956.417067
```

Figure 44: Output of the code

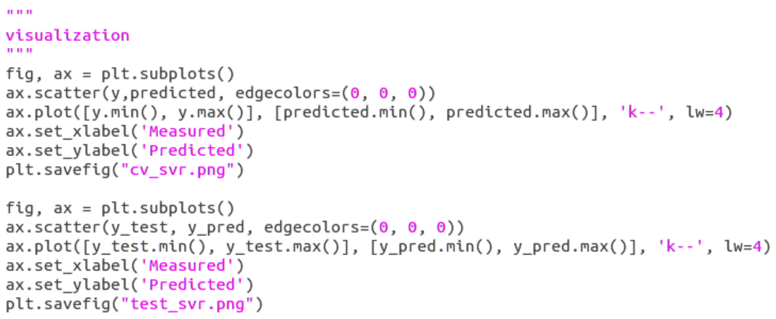
For easy understanding, the result is summarized in a table as below:



SVR	MVS	R2	MSE
Cross validation	0.5753	0.5485	-1005003
Testing Score	0.7201	0.7187	706956

Figure 45: testing scores of the SVR model

In order to understand how well the prediction result is, a plotting is made by taking the measured values as x-axis, and predicted values as y-axis. The code for the plotting is:



```
"""
visualization
"""
fig, ax = plt.subplots()
ax.scatter(y,predicted, edgecolors=(0, 0, 0))
ax.plot([y.min(), y.max()], [predicted.min(), predicted.max()], 'k--', lw=4)
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
plt.savefig("cv_svr.png")

fig, ax = plt.subplots()
ax.scatter(y_test, y_pred, edgecolors=(0, 0, 0))
ax.plot([y_test.min(), y_test.max()], [y_pred.min(), y_pred.max()], 'k--', lw=4)
ax.set_xlabel('Measured')
ax.set_ylabel('Predicted')
plt.savefig("test_svr.png")
```

Figure 46: Code for visualization of predicting result of SVR model

The figure below shows the visualization result of the cross validation:

The result of the visualization result of testing SVR model is as the figure below:

From the visualization result, we can conclude that the prediction for low value of PV generation is more accurate than the high value.

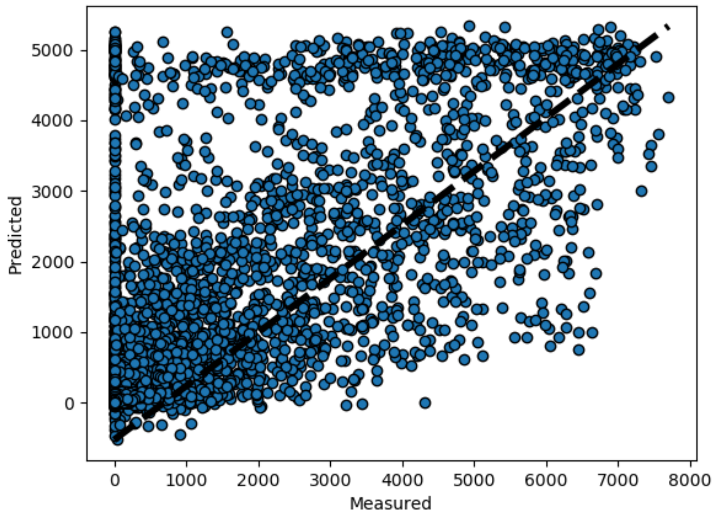


Figure 47: Visualization of predicting result of SVR model

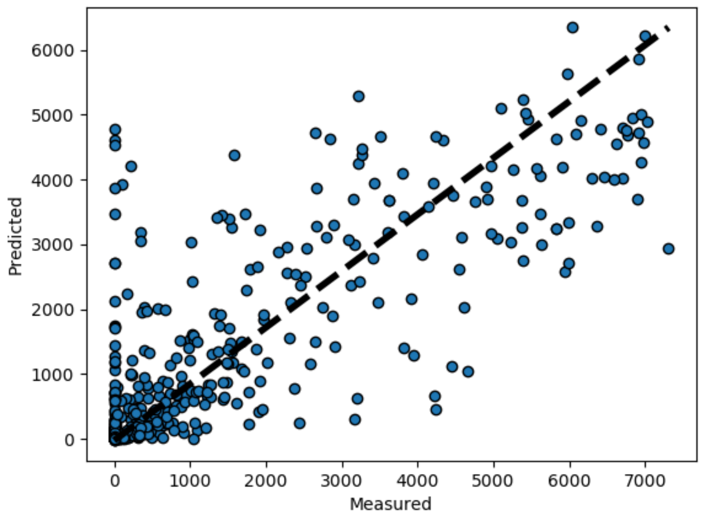


Figure 48: Visualization of predicting result of SVR model

5.3 MLP model

5.3.1 Principle of MLP

MLP(Multi-layer Perceptron) is a deep learning algorithm which is also named ANN(Artificial Neural Networks). This algorithm is inspired by the function of neurons in biology. Multi-layer feed-forward neural networks is one of the artificial neural networks. It consists of one input layer, one or more hidden layer and one output layer. Each layer is made up by units which is also called neurons.The structure is illustrated in the figure below:

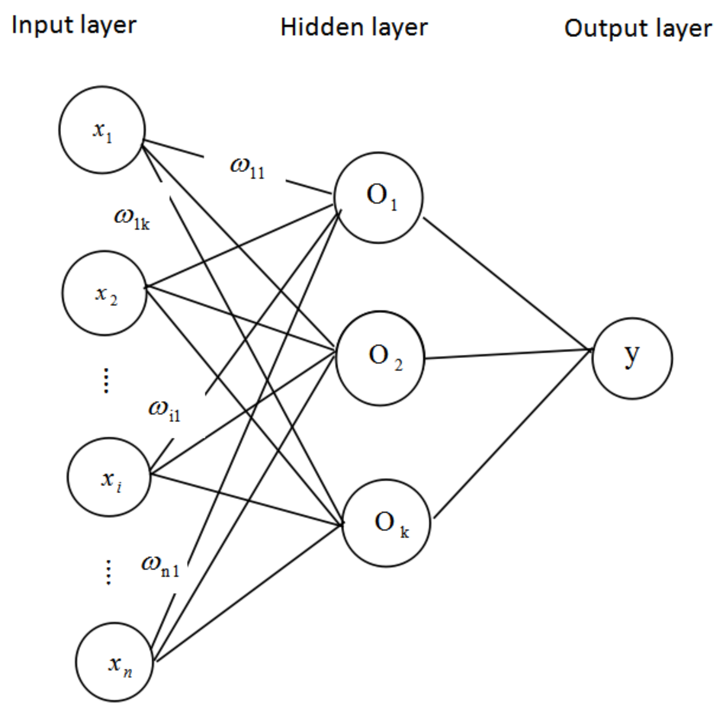


Figure 49: Structure of MLP algorithm

Input layer is used for input of data samples used for training. $x_1, x_2, ..., x_i, ..., x_n$ representing the features of the dataset. The number of hidden layer in this figure is one, actually, the number of hidden layers could be more than one, and the neurons of hidden layer is arbitrary, it should be adjusted according the characteristics of the specific problem and evaluation result of the performance. The neurons in the hidden layer transform the neurons in previous layer by computing the sum of the weighted linear combination of the values from previous layer. For example, in the figure, $O_1 = \omega_1 x_1 + \omega_2 x_2 + ... + \omega_i x_i +, ..., +\omega_n x_n$. after a linear transformation, a non-linear activation function is taken to transform the value again. Part of the neurons will be activated by the activation function. The output layer receives the values from the last hidden later, and then transform them to output values. One of the advantage of MLP is that it can learn a non-linear model. While the disadvantage of MLP is that many parameters need to be set to get good performance such as the number of hidden layers, the number of neurons in hidden layer, and the number of iterations. In addition, MLP is sensitive to data scaling, so standardization is required before using MLP algorithm. MLP algorithm can be used to solve both classification problem and also regression problem. In scikit-learn library, the class $sklearn.neural\_network.MLPRegressor$ provides the implementation of MLP regression function, and the $sklearn.neural\_network.MLPClassifier$ provides the MLP classification function for solving classification problem.

5.3.2 Parameter selection

Activation function is a crucial parameter in MLP model. There are three activation function are usually used in MLP. Sigmoid function(also called logistic function), tanh function and ReLU function[38]. The expression of Sigmoid is :

$$f(x) = \frac{1}{1+e^{-x}}$$

The output of Sigmoid is in $(0, 1)$, the lotting of Sigmoid is as the figure below:

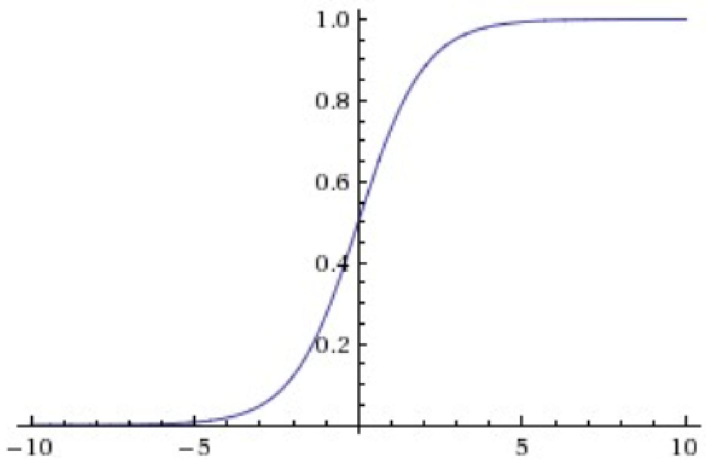


Figure 50: Plotting of Sigmoid

The advantage of sigmoid is: The output out this function is mapped into a limited range, so the optimization is stable, and it is easy to derivate. The disadvantage is that the output of this function is not centered on 0.

The tanh function is also one of the widely used activation function, the expression of the function is :

$$tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$

The plotting of this function is: The output of tanh function is in $[-1, 1]$,

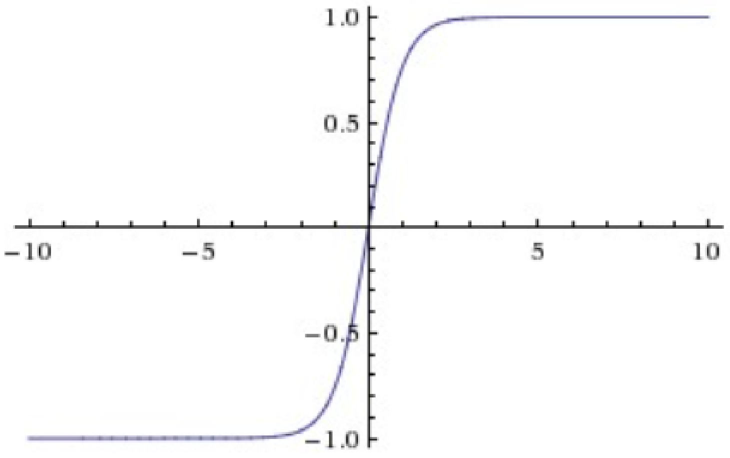


Figure 51: Plotting of tanh

and centered on 0. Tanh function get convergence faster than Sigmoid.

ReLU function is defined as

$$y = \begin{cases} 0 & (x \leq 0) \\ x & (x > 0) \end{cases}$$

The plotting of ReLU funstion is :

The weight optimizer is another important parameter in building a MLP regression model.In *sklearn.neural_network.MLPRegressor*, it is taken as the second parameter *solver*. Scikit-learn library provides three algorithms for weight optimization. One is 'lbfgs', it is an optimizer in the family of quasi-Newton methods. This method is better for small datasets in the respect of converge speed and the performance. Stochastic gradient descent algorithm is referred as 'sgd', while 'adam' refers to another optimizer based on stochastic gradient descent algorithm. This is proposed by Kingma, Diederik, and Jimmy Ba. The default *solver* in *sklearn.neural_network.MLPRegressor* is 'adam', it performs pretty

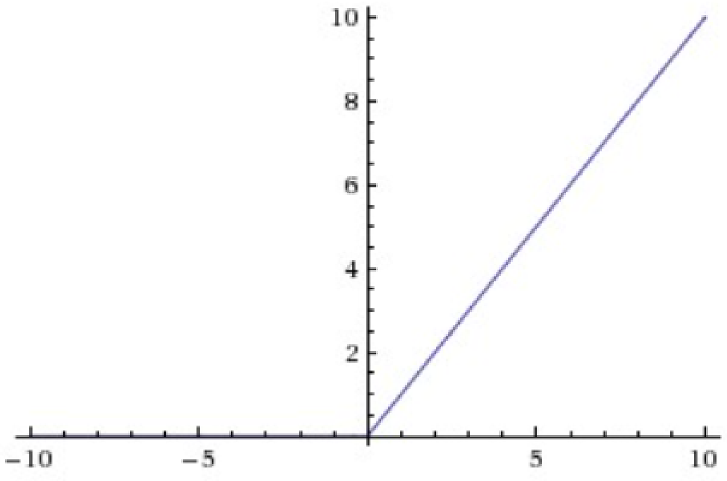


Figure 52: Plotting of ReLU

well for relative large datasets, such as thousands of training samples or more. It can make pretty good result in both training time and validation scores.

For the parameter *hidden_layer_size*,it is suggested one hidden layer is enough for normal case. As for the number of neurons in hidden layer, some experimental formula could be used in estimating the best number of neurons, Chow[39] proposed a calculation method by:

$$H_n = \frac{I_n + O_n}{2} + \sqrt{S_n}$$

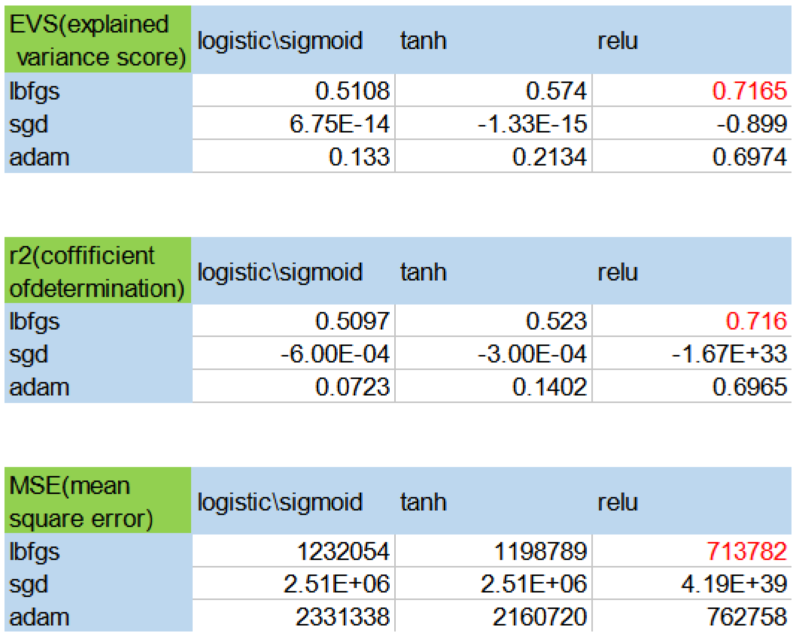
H_n is the estimated number of nodes in hidden layer, I_n is the number of features of input layer, O_n is the number of variables in output layer S_n is the number of data samples.

In this case, the number of data sample is 8760, so the number of neurons of input layer is 6, the number of neuron in the output layer is 1. Thus, computed according to the three formula, the best number of neurons in the hidden layer is around 100.

Based on the estimation of the number of neurons in the hidden layers, a test on looking for the best parameters of the activation function(*activation*) and optimizer(*solver*) is performed. (ESV)Explained variance score, R2 score (R^2, coefficient of determination), MSE(Mean Square Error) are taken as the measurements. The result of the testing is as the figure below:

The result of the testing indicates that the combination for 'relu' and 'lbfgs' is has the best performance.

Based on the best parameter combination we get for the *activation* and *solver*, another test is performed for verifying the parameter $hidden_layer_size$.



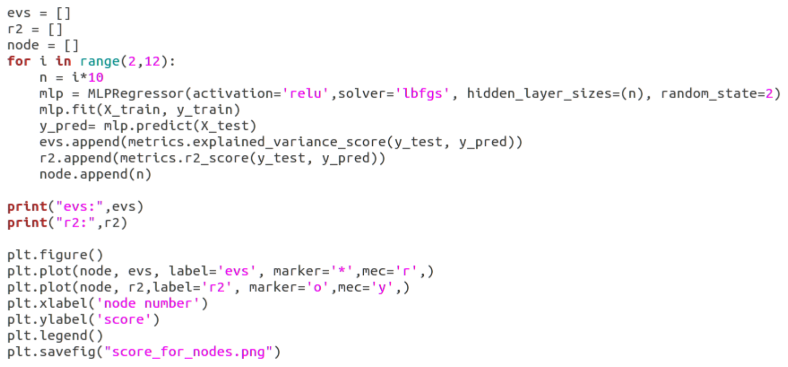
EVS(explained variance score)	logistic\sigmoid	tanh	relu
lbfgs	0.5108	0.574	0.7165
sgd	6.75E-14	-1.33E-15	-0.899
adam	0.133	0.2134	0.6974

r2(coffificient ofdetermination)	logistic\sigmoid	tanh	relu
lbfgs	0.5097	0.523	0.716
sgd	-6.00E-04	-3.00E-04	-1.67E+33
adam	0.0723	0.1402	0.6965

MSE(mean square error)	logistic\sigmoid	tanh	relu
lbfgs	1232054	1198789	713782
sgd	2.51E+06	2.51E+06	4.19E+39
adam	2331338	2160720	762758

Figure 53: Result for selecting parameter *activation* and *solver*

the measurements adopted are (ESV)Explained variance score, R2 score (R^2, coefficient of determination) in this verification process.Part of the code for the verification is shown in the figure below:



```
evs = []
r2 = []
node = []
for i in range(2,12):
    n = i*10
    mlp = MLPRegressor(activation='relu',solver='lbfgs', hidden_layer_sizes=(n), random_state=2)
    mlp.fit(X_train, y_train)
    y_pred= mlp.predict(X_test)
    evs.append(metrics.explained_variance_score(y_test, y_pred))
    r2.append(metrics.r2_score(y_test, y_pred))
    node.append(n)

print("evs:",evs)
print("r2:",r2)

plt.figure()
plt.plot(node, evs, label='evs', marker='*',mec='r',)
plt.plot(node, r2,label='r2', marker='o',mec='y',)
plt.xlabel('node number')
plt.ylabel('score')
plt.legend()
plt.savefig("score_for_nodes.png")
```

Figure 54: Partial code for verification of hidden layer size

The result of the this testing is shown in a figure:

From the result we can see that 100 is one of the pretty good number for the hidden layer nodes, the node number of 70 performs even a little better. Except that, the node number of 50 also performs pretty good .

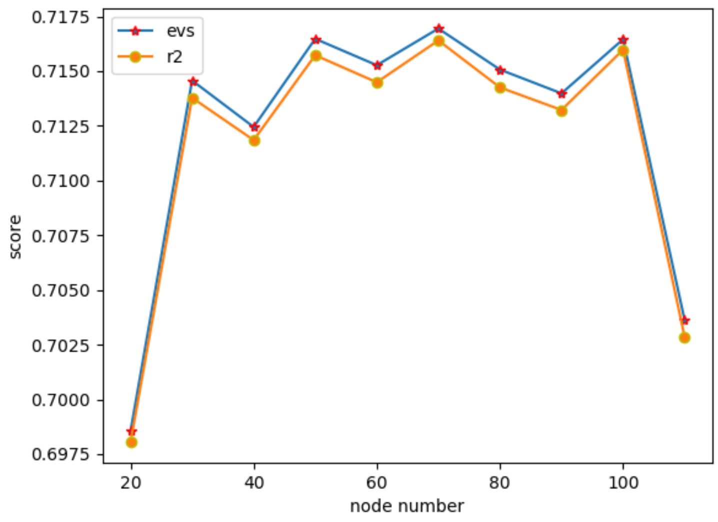


Figure 55: Scores of different hidden layer size

5.3.3 MLP model implementation

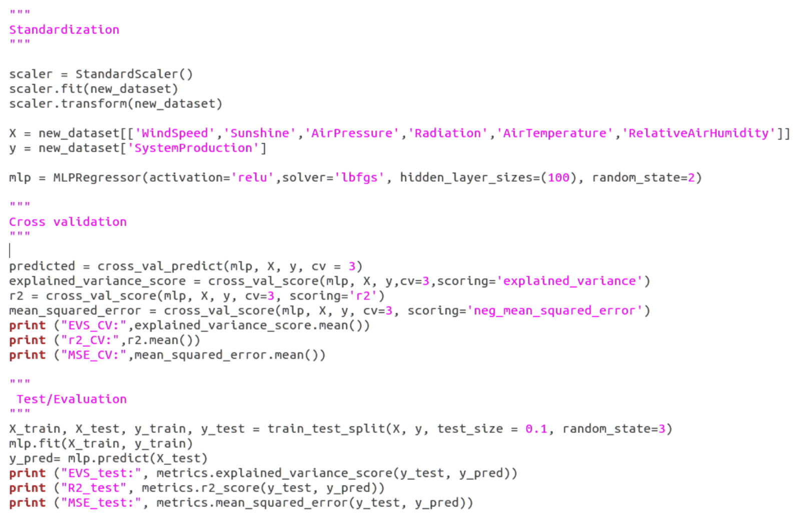
In the implementation of MLP model, a standardization is important before training the model because MLP model is sensitive to the scale of data. By using the parameters selected in the last section, the MLP is fixed.

In testing the result of the implementation of MLP model, the data set is split into training data set and testing data set. Two splitting methods are used here. First one is cross validation, 3-fold cross validation is used in this case, and another splitting method is 'hold-out', 90% of the original data is taken for training data set and the rest 10% data is used as testing data set. In order to make comparison with other models, the random state of the 'hold-out' splitting is set at the same number 3 as in the previous model. Part of the code of the implementation is as below:

The same measurements are adopted as in linear model and SVR model, MVS(Explained variance Score), R2 score (R^2, coefficient of determination), MSE(Mean Square Error). The output of the code gives the scores of this three measurement for cross validation and testing respectively:

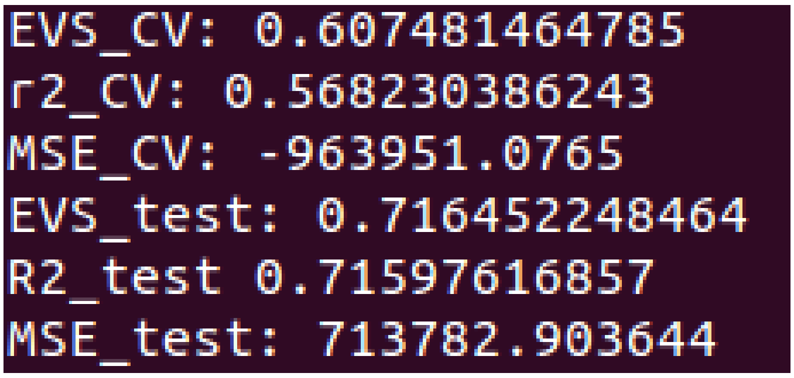
Summarizing the result in a table is :

In order to see the distribution of the prediction, and compare it with the distribution of measured data. Two visualization are given for both the result of cross validation and testing. The figure below is the visualization for the result cross validation:



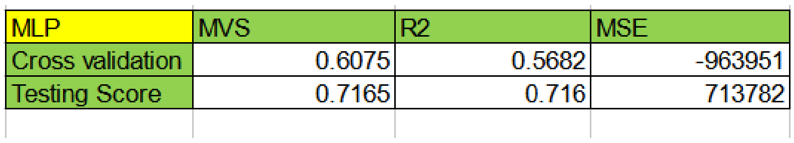
```
"""
Standardization
"""

scaler = StandardScaler()
scaler.fit(new_dataset)
scaler.transform(new_dataset)

X = new_dataset[['WindSpeed','Sunshine','AirPressure','Radiation','AirTemperature','RelativeAirHumidity']]
y = new_dataset['SystemProduction']

mlp = MLPRegressor(activation='relu',solver='lbfgs', hidden_layer_sizes=(100), random_state=2)

"""
Cross validation
"""

predicted = cross_val_predict(mlp, X, y, cv = 3)
explained_variance_score = cross_val_score(mlp, X, y,cv=3,scoring='explained_variance')
r2 = cross_val_score(mlp, X, y, cv=3, scoring='r2')
mean_squared_error = cross_val_score(mlp, X, y, cv=3, scoring='neg_mean_squared_error')
print ("EVS_CV:",explained_variance_score.mean())
print ("r2_CV:",r2.mean())
print ("MSE_CV:",mean_squared_error.mean())

"""
 Test/Evaluation
"""
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state=3)
mlp.fit(X_train, y_train)
y_pred= mlp.predict(X_test)
print ("EVS_test:", metrics.explained_variance_score(y_test, y_pred))
print ("R2_test", metrics.r2_score(y_test, y_pred))
print ("MSE_test:", metrics.mean_squared_error(y_test, y_pred))
```

Figure 56: Partial code of MLP model



```
EVS_CV: 0.607481464785
r2_CV: 0.568230386243
MSE_CV: -963951.0765
EVS_test: 0.716452248464
R2_test 0.71597616857
MSE_test: 713782.903644
```

Figure 57: Output of the code of MLP model implementation



MLP	MVS	R2	MSE
Cross validation	0.6075	0.5682	-963951
Testing Score	0.7165	0.716	713782

Figure 58: Scores of MLP model

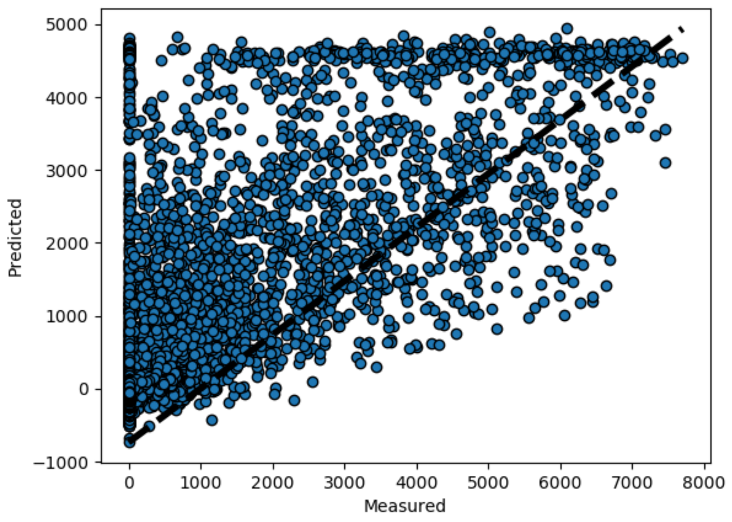


Figure 59: Visualization of cross validation for MLP model

From the result of the visualization, we notice that the distribution is quite different with that in linear and SVR model. The predicted value of the PV generation could be a negative value. The visualization for the testing result of MLP is shown in the next figure:

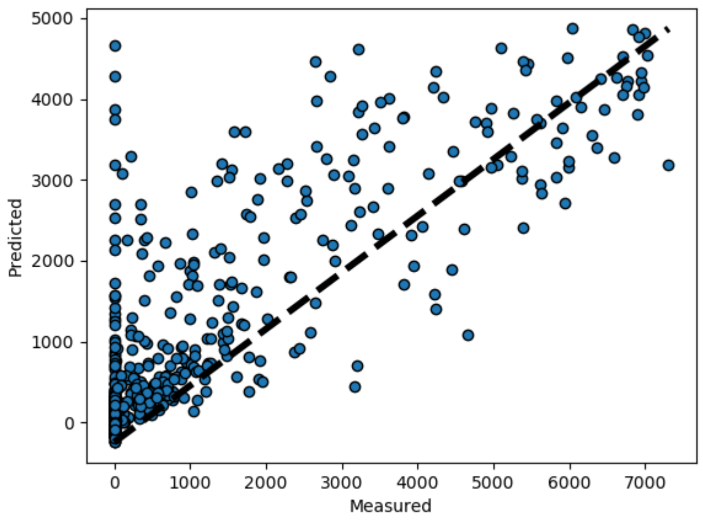


Figure 60: Visualization of testing for MLP model

6 Model Evaluation

6.1 Evaluation measurement

For the purpose of evaluating the performance of learning models, many different measurements are adopted. In scikit-learn library, a metrics module is provided for performing the evaluation of machine learning models.

The metrics module includes classification metrics, regression metrics, multi-label ranking metrics, clustering metrics, biclustering metrics and pariewise metrics. For the problem is a regression problem, so regression metrics would be used for model evaluation. The regression metrics provide several score,loss, and utility functions to measure performance of regression model.

6.1.1 Explained variance score

The $explained\_variance\_score$ function computes the explained variance regression score. The return result is the explained variance or ndarray if $'multioutput'$ is $'raw_values'$. The explained variance is calculated as[40]:

$$explained\_variance(y, \hat{y}) = 1 - \frac{Var\{y - \hat{y}\}}{Var\{y\}}$$

In this expression, $\hat{y}$ is the estimated of the target output, and y is the corresponding real target output, and Var represents the Variance, which means the square of the standard deviation. The best possible result of the estimation is 1.0, the lower values, the worse the estimation.

6.1.2 Mean absolute error

Mean absolute error is computed by the function $mean\_absolute\_error$. It is a metric which is corresponding to the expected value of the absolute error or $l1 - norm$ loss. The output of this function is non-negative floating point. The best value is 0. The higher result indicates the worse performance. MAE is defined as[40]:

$$MAE(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} |y_i - \hat{y}_i|$$

In this expression, $\hat{y_i}$ represents the estimated value of the sample i, and y_i is the corresponding correct output of sample i. n is the number of data samples.The mean absolute error(MAE) is estimated over n samples.

6.1.3 Mean squared error

The function *mean_absolute_error* in scikit-learn computes mean square error, it is a metric corresponding to the expected value of the squared error or loss. the return result is a non-negative floating point value(the best result is 0.0), or an array of floating point values, one for each individual target.MSE is defined as[40]:

$$MSE(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2$$

In this expression, $\hat{y}_i$ represents the estimated value of the sample i, and y_i is the corresponding correct output of sample i. n is the number of data samples.The mean squared error(MSE) is estimated over n samples.

6.1.4 Mean squared logarithmic error

Mean square logarithmic error is computed by the function *mean_squared_log_error* in scikit-learn library. It computes a risk metric which is corresponding to the expected value of the squared error or loss. The return result of this function is a non-negative floating point value(The best result is 0.0),or an array of floating point values, one for each individual target. MSLE is defined as[40]:

$$MSLE(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1}(log_e(y_i + 1) - log_e(\hat{y}_i + 1))^2$$

In this expression, $\hat{y}_i$ represents the estimated value of the sample i, and y_i is the corresponding correct value of sample i. n is the number of data samples.The mean squared logarithmic error(MSLE) is estimated over n samples.

6.1.5 Median absolute error

The $median\_absolute_error$ function in scikit-learn library computes median absolute error. It is robust to outliers.The loss is calculated by calculating the median of all absolute differences between the target and the prediction. The return result of this function is a positive floating point value, the best possible result is 0.0. The median absolute error(MedAE) is defined as[40]:

$$MedAE(y, \hat{y} = median(|y_1 - \hat{y}_1|, ...|y_i - \hat{y}_i|, ..., |y_n - \hat{y}_n|)$$

In this expression, $\hat{y}_i$ represents the estimated value of the sample i, and y_i is the corresponding true value of sample i. n is the number of data samples.The median absolute error(MedAE) is estimated over n samples.

6.1.6 R^2 score

The coefficient of determination is also called R^2. It is computed by the function $r2\_score$ in scikit-learn library. This function provides a measurement of how well in the future the new samples could be predicted by this model.The best result of the score is 1.0, and it can be a negative value, because the estimated model could be arbitrary worse. For a constant model, which always predicts the expected value o y, disregarding the input features, this would get a R^2 score of 0.0. The expression of R^2 [40]is:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \hat{y_i})^2}{\sum_{i=0}^{n-1}(y_i - \bar{y_i})^2}$$

In this expression, $\hat{y_i}$ represents the estimated value of the sample i, and y_i is the corresponding true value of sample i. n is the number of data samples. The score R^2 is estimated over n samples.

6.2 Evaluation result

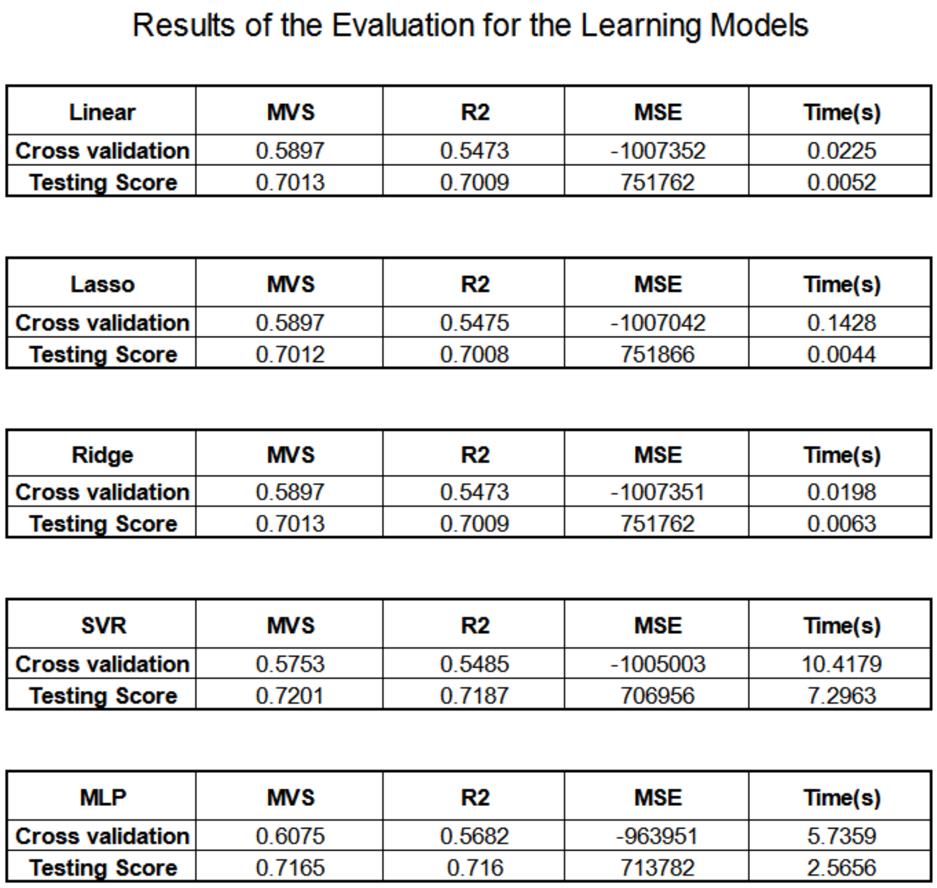
The learning models built in the last section are evaluated by the measurement of explained variance score(EVS), mean square error(MSE), and R^2 score(R2). Both cross validation result and testing result are evaluated by the measurements. In addition, the processing time for cross validation and testing are also measured. The results of the evaluation for the five models (linear,lasso, ridge, SVR and MLP) are shown in the table below:

From the summary of the evaluation result some conclusion we can get:

• by the measurement of MVS, R2 and MSE, the best scores of cross validation is the MLP model, and followed by the SVR model, while the Linear regression model, Lasso regression model and Ridge regression model get almost the same result which get a little less score than that SVR model and MLP model;

• As for the evaluation result for the testing period, the SVR model performs best and second is MLP model. The result for the Linear regression model, Lasso regression model and Ridge regression model are also similar in test period, and perform a little worse than SVR model and MLP model.

• In term of processing time, the Linear regression model, Lasso regression model and Ridge regression model perform best, for the cross validation period, Lasso regression model spends a little longer time than Linear regression model and Ridge regression model, but not big difference for the time spent in testing period. However, The time spent for



Results of the Evaluation for the Learning Models

Linear	MVS	R2	MSE	Time(s)
Cross validation	0.5897	0.5473	-1007352	0.0225
Testing Score	0.7013	0.7009	751762	0.0052

Lasso	MVS	R2	MSE	Time(s)
Cross validation	0.5897	0.5475	-1007042	0.1428
Testing Score	0.7012	0.7008	751866	0.0044

Ridge	MVS	R2	MSE	Time(s)
Cross validation	0.5897	0.5473	-1007351	0.0198
Testing Score	0.7013	0.7009	751762	0.0063

SVR	MVS	R2	MSE	Time(s)
Cross validation	0.5753	0.5485	-1005003	10.4179
Testing Score	0.7201	0.7187	706956	7.2963

MLP	MVS	R2	MSE	Time(s)
Cross validation	0.6075	0.5682	-963951	5.7359
Testing Score	0.7165	0.716	713782	2.5656

Figure 61: Evaluation results for different learning models

SVR model is quite long, more than 10 seconds is used in the process of cross validation, and around 7 seconds for testing process. As for MLP model, it is also a little time consuming in comparison with Linear regression model, Lasso regression model and Ridge regression model, in the process of cross validation, about 5.7 seconds was used. For the testing period, the time spent was about 2.5 second, which is a lot better than SVR model.

In summary, for the Linear regression model, Lasso regression model and Ridge regression model, the short processing time for training and testing is the advantage, but compromising a little on accuracy of prediction. MLP model is a better choice in considering both the accuracy and processing time for training and testing data.

7 Real-time monitoring

As it is described in the first section, the key work in real-time monitoring is to predict the PV power generation. Since the PV generation forecasting model has been implemented and tested in Section 5, and the evaluation is performed in Section 6. The last step in the process is to compare the PV power generation with the predicted value of PV power generation at the same point.

Since the models are trained based on history weather data and PV power generation in 1 hour interval. When these forecasting model is taken into utilization, the forecasting weather data is required in advance.

The location of weather forecasting should be the same as the history weather data measured in the training process. The variables of the weather forecasting data should be the same as they are used in the training model process, and they are expected to be in 1 hour interval as well.

As far as the forecasting weather data is provided, the prediction of the PV power generation will be obtained by utilizing the forecasting model proposed in previous section. So that the comparison can be maintained by computing the difference between the predicted PV power generation value with the PV power generation value generated in real-time.

If the difference between the two values is bigger than a benchmark which can be given by the users. that will be taken as a bad value, if the state lasts for a specific period of time(such as 5 hours) represented by n, it could also be given by the users, then an 'alert' should be triggered;

If the difference between the two values is not bigger than the benchmark, a 'normal state' will be the output. The process is shown in the next figure:

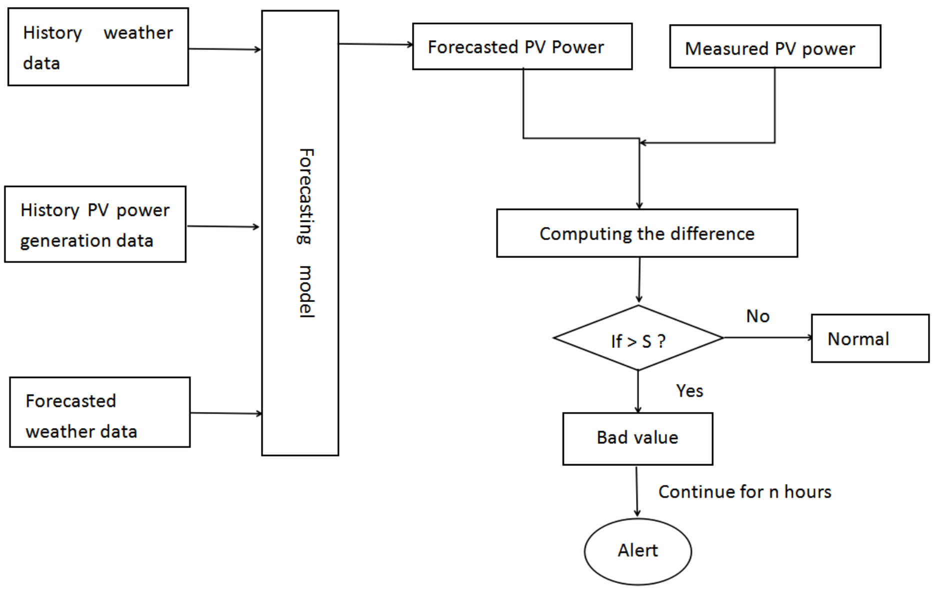


Figure 62: Process of PV system real-time monitoring

8 Conclusion

The main objective of this thesis is to propose a solution for monitoring PV system in real-time from the perspective of big data. Techniques related with solving big data issue is adopted including machine learning and some tools are utilized in implementing the PV generation forecasting model such as python language, numpy library, scikit learn library and matplot library.

Data mining methods adopted in solving big data problem in this thesis include data collection, data preprocessing, data standardization, data set splitting and feature selection. In the process of feature selection, three methods are taken and compared in order to increase the reliability.

Upon the result of data mining, five forecasting models are built based on machine learning algorithms and deep learning algorithms. The five forecasting models are: Linear regression model, Lasso regression model, Ridge regression model,SVR model and MLP model. In the selection of parameter for SVR model, and MLP model, various test are taken with visualization showing the testing result.

In order to view the result of the forecasting, visualization of the forecasting result of the five models is performed.

For the purpose of evaluating the performance of each forecasting model and making comparison with each other, various of measurements are adopted in evaluation of the performance of the five forecasting models. The measurements are explained variance score (EVS) mean square error(MSE), and R^2 score(R2), except that, processing time is taken as a measurement as well.

For making sure the evaluation result is not fooled by randomness, two types of data splitting methods are adopted. One is cross validation, and another is 'hold-out' by setting a same random state of all the forecasting models.

The contribution of this thesis include:

- Provide a practical solution for PV system real-time monitoring.
- Analyzing energy big data by the means of data mining, machine learning and deep learning.
- Implement five forecasting model to predict the PV power generation.
- Visualization the testing result of the five forecasting model
- Evaluate the performance of the five forecasting models, with the comparison of advantages and disadvantages of each models.

9 Discussion

Though big data, machine learning are not brand new techniques nowadays, and many researchers have proposed forecasting models for PV power generation based on artificial neural networks or SVR which obtained good results recently.

However, in this thesis, the achievement is not only on proposing forecasting models for PV generation based on big data, machine learning techniques, but also a solution for PV system real-time monitoring, especially for the those location with extreme weather conditions like Norway which is one of the northernmost countries. Many research investigate PV generation with data collected from tropical area, but not much research has been found investigating PV generation for high latitude area. Due to the high latitude, the sunlight is extremely little in winter, while in summer, sunlight is extremely much.

What is not satisfying enough in this thesis is the accuracy of the forecasting result . As it is known, the accuracy of the prediction for PV power generation is greatly depend on the weather data. In this thesis, the history weather data are collected from an observation station at Blindern, Oslo, which is the closest observation station to IFE located at Kjeller, Oslo. But it is still 25 Kilometers distance between Blindern and Kjeller Oslo. Thus, the accuracy of the history weather data is not good enough for predicting the PV power generation in Kjeller. This is supposed to be the main reason for the compromised forecasting accuracy. Therefore, it is significant work to improve the accuracy of weather forecasting in order to get better prediction for PV power generation.

In addition, more attention should be paid to analyzing data before training the data by using algorithms, such as distinguishing false values and remove them, and there is still a lot of methods need to be discovered in order to improve the accuracy of forecasting models.

At last, the algorithm utilized in forecasting PV power generation is expected be optimized to improve the performance of forecasting models, and more techniques related with big data could be taken into consideration for PV system real-time monitoring in the future.

10 Acknowledgements

I would like to show great appreciate to my supervisor Yan Zhang, who has given direction and suggestions for this thesis, and also thanks to Junjie zhu, Erik Stensrud Marstein, Tor-Jørgen Ervum, and Mari Øgaard for the data collecting work. At last, I would like to thank my family, this thesis cannot be accomplished without their support.

11 Appendix

The data for investigating in this thesis, and the code for implementing the forecasting model, and visualization of the result and so on can all be found on this website:

https://github.com/liulu1991/Big-data-analytics-for-PV-system-real-time-monitoring

References

[1] Adolf Goetzberger and Volker Uwe Hoffmann. *Photovoltaic solar energy generation*, volume 112. Springer Science & Business Media, 2005.

[2] PVPS IEA-PVPS. Report snapshot of global pv 1992-2014. *Report IEA-PVPS T1-26*, 2015.

[3] Jie Shi, Wei-Jen Lee, Yongqian Liu, Yongping Yang, and Peng Wang. Forecasting power output of photovoltaic systems based on weather classification and support vector machines. *IEEE Transactions on Industry Applications*, 48(3):1064–1069, 2012.

[4] Jabar H Yousif, Hussein A Kazem, and John Boland. Predictive models for photovoltaic electricity production in hot weather conditions. *Energies*, 10(7):971, 2017.

[5] Changsong Chen, Shanxu Duan, Tao Cai, and Bangyin Liu. Online 24-h solar power forecasting based on weather type classification using artificial neural network. *Solar Energy*, 85(11):2856–2870, 2011.

[6] Hussein A Kazem, Jabar H Yousif, and Miqdam T Chaichan. Modeling of daily solar energy system prediction using support vector machine for oman. *International Journal of Applied Engineering Research*, 11(20):10166–10172, 2016.

[7] Jun Liu, Wanliang Fang, Xudong Zhang, and Chunxiang Yang. An improved photovoltaic power forecasting model with the assistance of aerosol index data. *IEEE Transactions on Sustainable Energy*, 6(2):434–442, 2015.

[8] JH Yousif and Hussein A Kazem. Modeling of daily solar energy system prediction using soft computing methods for oman. *Research Journal of Applied Sciences, Engineering and Technology*, 13(3):237–244, 2016.

[9] Hussein A Kazem and Jabar H Yousif. Comparison of prediction methods of photovoltaic power system production using a measured dataset. *Energy Conversion and Management*, 148:1070–1081, 2017.

[10] Tamer Khatib, Azah Mohamed, M Mahmoud, and K Sopian. Estimating global solar energy using multilayer perception artificial neural network. *International journal of energy*, 6(1):82–87, 2012.

[11] Ozan Şenkal and Tuncay Kuleli. Estimation of solar radiation over turkey using artificial neural network and satellite data. *Applied Energy*, 86(7-8):1222–1228, 2009.

[12] Amit Kumar Yadav and SS Chandel. Identification of relevant input variables for prediction of 1-minute time-step photovoltaic module power using artificial neural network and multiple linear regression models. *Renewable and Sustainable Energy Reviews*, 77:955–969, 2017.

[13] Yingni Jiang. Prediction of monthly mean daily diffuse solar radiation using artificial neural networks and comparison with other empirical models. *Energy policy*, 36(10):3833–3837, 2008.

[14] Barnabas K Tannahill and Mo Jamshidi. System of systems and big data analytics–bridging the gap. *Computers & Electrical Engineering*, 40(1):2–15, 2014.

[15] Atsu SS Dorvlo, Joseph A Jervase, and Ali Al-Lawati. Solar radiation estimation using artificial neural networks. *Applied Energy*, 71(4):307–319, 2002.

[16] MA Behrang, E Assareh, A Ghanbarzadeh, and AR Noghrehabadi. The potential of different artificial neural network (ann) techniques in daily global solar radiation modeling based on meteorological data. *Solar Energy*, 84(8):1468–1480, 2010.

[17] Sue Ellen Haupt and Branko Kosovic. Big data and machine learning for applied weather forecasts: Forecasting solar power for utility operations. In *Computational Intelligence, 2015 IEEE Symposium Series on*, pages 496–501. IEEE, 2015.

[18] Panagiotis D Diamantoulakis, Vasileios M Kapinas, and George K Karagiannidis. Big data analytics for dynamic energy management in smart grids. *Big Data Research*, 2(3):94–101, 2015.

[19] Big data: The 5 vs everyone must know. https://www.linkedin.com/pulse/20140306073407-64875646-big-data-the-5-vs-everyone-must-know/.

[20] Wei Fan and Albert Bifet. Mining big data: current status, and forecast to the future. *ACM sIGKDD Explorations Newsletter*, 14(2):1–5, 2013.

[21] Albert Bifet. Mining big data in real time. *Informatica*, 37(1):15, 2013.

[22] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[23] Nicolas Le Roux, Yoshua Bengio, and Andrew Fitzgibbon. 15 improving first and second-order methods by modeling uncertainty. *Optimization for Machine Learning*, page 403, 2011.

[24] Chris Stauffer and W. Eric L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):747–757, 2000.

[25] Soumen Chakrabarti, Martin Ester, Usama Fayyad, Johannes Gehrke, Jiawei Han, Shinichi Morishita, Gregory Piatetsky-Shapiro, and Wei Wang. Data mining curriculum: A proposal (version 1.0). *Intensive Working Group of ACM SIGKDD Curriculum Committee*, 140, 2006.

[26] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[27] David J Hand. Principles of data mining. *Drug safety*, 30(7):621–622, 2007.

[28] D Hand, Heikki Mannila, and Padhraic Smyth. Principles of data mining, cambridge, massachussets, 2001.

[29] python. https://docs.python.org/3/tutorial/.

[30] pandas. http://pandas.pydata.org/.

[31] pandas. http://pandas.pydata.org/pandas-docs/stable/.

[32] numpy. http://www.numpy.org/.

[33] Zhihua Zhou. *machine learning*. Qing hua da xue chu ban she, 2016.

[34] David N. Reshef, Yakir A. Reshef, Hilary K. Finucane, Sharon R. Grossman, Gilean Mcvean, Peter J. Turnbaugh, Eric S. Lander, Michael Mitzenmacher, and Pardis C. Sabeti. Detecting novel associations in large data sets. *Science*, 334(6062):1518, 2011.

[35] Lassoregression. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html#sklearn.linear_model.Lasso.

[36] Ridgeregression. http://scikit-learn.org/stable/modules/linear_model.html#ridge-regression.

[37] Support vector machine. http://scikit-learn.org/stable/modules/svm.html#svm.

[38] Mlpregressor. http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html#sklearn.neural_network.MLPRegressor.

[39] Stanley KH Chow, Eric WM Lee, and Danny HW Li. Short-term prediction of photovoltaic energy generation by intelligent approach. *Energy and Buildings*, 55:660–667, 2012.

[40] metrics. http://scikit-learn.org/stable/modules/model_evaluation.html#explained-variance-score.