SML 2010

Name : Gautam Mishra

Reg. No. : 12010046

ABSTRACT:

The area of machine learning experienced tremendous growth in 2010, during which time new methods and algorithms were created and used in a variety of applications. Support Vector Machines (SVMs), Random Forest, K-Nearest Neighbors, Naive Bayes, and Gradient Boosting Machines are just a few of the common machine learning methods covered in this study (GBMs). I examined the studies from this era that show how these algorithms perform, are applicable, and have limits in a variety of fields, including text categorization, intrusion detection, and land use classification. I have also emphasised the significance of choosing the appropriate algorithm for a particular task and the potential advantages of mixing various algorithms to increase forecast accuracy.

KEYWORDS :

Random Forest , regression , gradient , function , decision tree , libraries , packages

INTRODUCTION

2010 witnessed a tremendous rise in the field of machine learning, making it an important year. The most widely utilised machine learning techniques at the time included the following:

Support Vector Machines (SVMs): SVMs were already a well-liked technique in the early 2000s, but they gained more traction in 2010. SVMs are frequently used for classification issues, and they operate by locating a hyperplane that divides several classes in a dataset

Random Forest: Random Forest is an ensemble method that combines various decision trees to produce predictions. It is frequently employed for classification and regression issues.

K-Nearest Neighbors (KNN): KNN is a straightforward but efficient technique that is frequently used for classification issues. It functions by locating the k-nearest data points to a new observation and utilising their class labels to predict.

❖ What is regression?

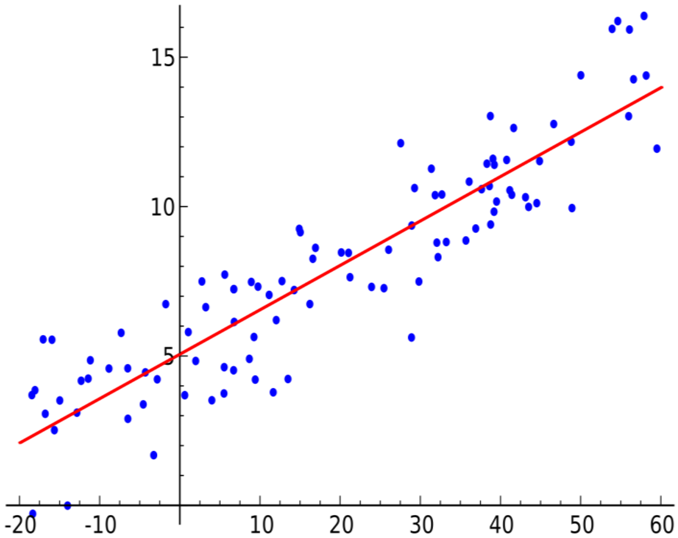
The primary objective of regression is the creation of an effective model to forecast the dependent characteristics from a variety of attribute variables. When the output variable, such as a salary, weight, area, or other continuous number, is either real or has a discrete value, a regression problem exists.

Regression is also a statistical tool that can be utilised in a variety of contexts, including finance and real estate. The link between a dependent variable and a number of independent factors is predicted using it. Examining several regression techniques, let's see what they are.

❖ What is linear regression?

The primary objective of regression is the creation of an effective model to forecast the dependent characteristics from a variety of attribute variables. When the output variable, such as a salary, weight, area, or other continuous number, is either real or has a discrete value, a regression problem exists.

Regression is also a statistical tool that can be utilised in a variety of contexts, including finance and real estate. The link between a dependent variable and a number of independent factors is predicted using it. Examining several regression techniques, let's see what they are.



A type of regression analysis known as simple linear regression involves only one independent variable and a linear connection between the independent (x) and dependent (y) variables. The best-fit straight line is indicated by the red line in the graph above. We attempt to draw a line from the provided data points that best represents the points. The linear equation below can be used to represent the line.

y = a_0 + a_1 * x ## Linear Equation

In order to identify the best values for a 0 and a 1, the linear regression algorithm seeks to maximise its efficiency. Two key ideas include :

❖ Cost Function

The optimal values for a 0 and a 1 that would produce the best fit line for the data points can be determined using the cost function. We transform this search problem into a minimization problem because we want and minimise the difference between the predicted value and the actual value since we want the best values for a 0

$$minimize \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

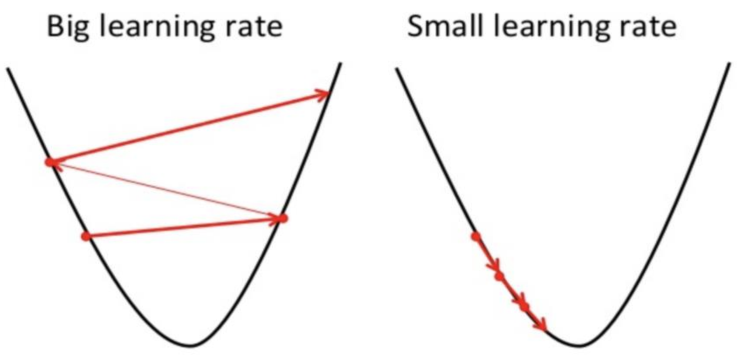
$$J = \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

a 1.

We decide to minimise the previous function. The error difference is determined by comparing the anticipated values to the actual values. The error difference is squared, added to the total number of data points, and then its square root is divided by the number of data points. The average squared error across all data points is shown in this. This cost function is also known as the Mean Squared Error (MSE) function. In order to get the MSE value to settle at the minima, we will now modify the values of a 0 and a 1 using this MSE function.

❖ Gradient Descent

Gradient descent is the next crucial idea that must be understood in order to comprehend linear regression. Update a 0 and a 1 using gradient descent to lower the cost function (MSE). Starting with given values for a 0 and a 1, the objective is to iteratively adjust these values to decrease the cost. We can adjust the values with the aid of gradient descent.



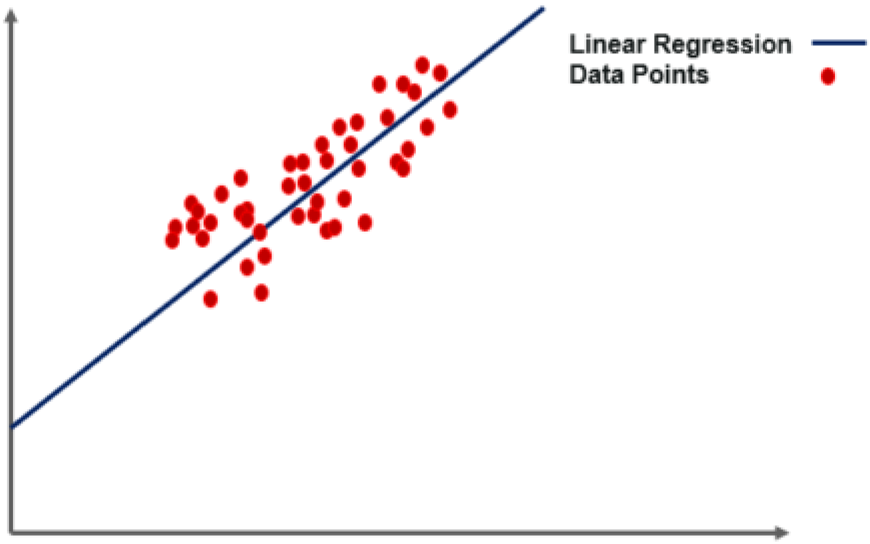
❖ Types Of Regression

Different types of regression are :

1. **Simple Linear Regression**
2. **Polynomial Regression**
3. **Support Vector Regression**
4. **Decision Tree Regression**
5. **Random Forest Regression**

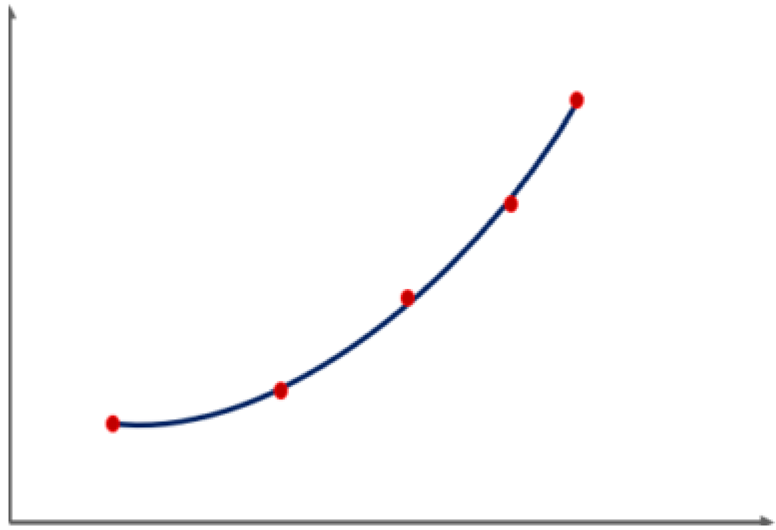
❖ Simple Linear Regression

Simple linear regression is one of the most interesting and popular regression techniques. In this case, a dependent variable's result is predicted using the independent variables; there is a linear relationship between the variables. the term "linear regression" was created.



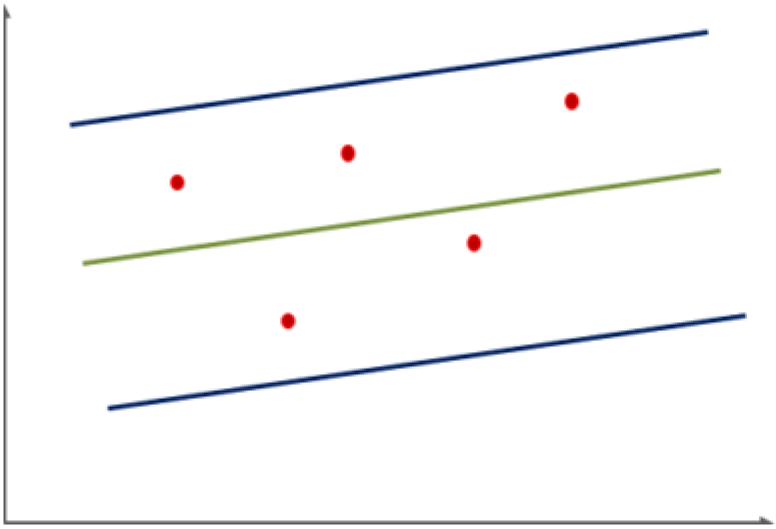
❖ Polynomial Regression

In this technique, we interchange the original features into polynomial features of a given degree and then the regression is performed on it.



❖ Support Vector Regression

In order to perform support vector machine regression, or SVR, we find a hyperplane with a maximum margin where the most data points can fit within the margins. It is very similar to the classification approach used with support vector machines..



❖ Decision Tree Regression

Regression and classification can both be done using a decision tree. Regression uses the ID3 Iterative Dichotomiser 3 (ID3) algorithm , which helps us find the splitting node by lowering the standard deviation.

❖ Random Forest Regression

We combine the predictions from various decision tree regressions in random forest regression . Examining Simple Linear Regression in details

❖ Advantages And Disadvantages

Advantages	Disadvantages
Linear regression performs exceptionally well for linearly separable data	The assumption of linearity between dependent and independent variables
Easier to implement, interpret and efficient to train	It is often quite prone to noise and overfitting
It handles overfitting pretty well using dimensionally reduction techniques, regularization, and cross-validation	Linear regression is quite sensitive to outliers
One more advantage is the extrapolation beyond a specific data set	It is prone to multicollinearity

Use Case – Implementing Linear Regression

The process takes place in the following steps:

1. **Importing Libraries and Packages**
2. **Loading and Viewing Data Set**
3. **Plotting and Visualizing Data**

4. Modeling and Predicting with sklearn

1. Importing Libraries and Packages

These programmes will be used to change the data, show the features and labels, and assess how well our model worked. The dataframe's columns and cells can be easily modified using Numpy and Pandas. Our data will be shown using Seaborn and matplotlib.

In [1]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

2. Loading and Viewing Data Set

Pandas enables us to load the training and test sets that we will need to train and validate our model. Look at our data table to examine the values we'll be using before we start so that we know what to expect. Examining some sample data and statistics is possible using the head and description functions.

In [2]:
```
df = pd.read_csv('NEW-DATA-1.T15.txt')
```

In [3]:
```
df
```

Out[3]:

	1:Date	2:Time	3:Temperature_Comedor_Sensor	4:Temperature_Habitacion_Sensor	5:Weather_Temperature	6:CO2_Comedor_Sensor	7:CO2_Habitacion_
0	13/03/2012	11:45	18.1875	17.8275	0.0000	216.560	
1	13/03/2012	12:00	18.4633	18.1207	6.8000	219.947	
2	13/03/2012	12:15	18.7673	18.4367	17.0000	219.403	
3	13/03/2012	12:30	19.0727	18.7513	18.0000	218.613	
4	13/03/2012	12:45	19.3721	19.0414	20.0000	217.714	
...	...	...	...	...	...	...	
2759	11/04/2012	05:30	21.1520	20.8187	13.0000	190.539	
2760	11/04/2012	05:45	21.0413	20.7053	12.1333	190.421	
2761	11/04/2012	06:00	20.9347	20.5827	12.0000	190.432	
2762	11/04/2012	06:15	20.8560	20.5200	12.0000	191.531	
2763	11/04/2012	06:30	20.7627	20.4400	12.1333	191.563	

2764 rows × 24 columns

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2764 entries, 0 to 2763
Data columns (total 24 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   1:Date                           2764 non-null   object
 1   2:Time                           2764 non-null   object
 2   3:Temperature_Comedor_Sensor     2764 non-null   float64
 3   4:Temperature_Habitacion_Sensor  2764 non-null   float64
 4   5:Weather_Temperature            2764 non-null   float64
 5   6:CO2_Comedor_Sensor             2764 non-null   float64
 6   7:CO2_Habitacion_Sensor          2764 non-null   float64
 7   8:Humedad_Comedor_Sensor         2764 non-null   float64
 8   9:Humedad_Habitacion_Sensor      2764 non-null   float64
 9   10:Lighting_Comedor_Sensor       2764 non-null   float64
 10  11:Lighting_Habitacion_Sensor    2764 non-null   float64
 11  12:Precipitacion                 2764 non-null   float64
 12  13:Meteo_Exterior_Crepusculo     2764 non-null   float64
 13  14:Meteo_Exterior_Viento         2764 non-null   float64
 14  15:Meteo_Exterior_Sol_Oest       2764 non-null   float64
 15  16:Meteo_Exterior_Sol_Est        2764 non-null   float64
 16  17:Meteo_Exterior_Sol_Sud        2764 non-null   float64
 17  18:Meteo_Exterior_Piranometro    2764 non-null   float64
 18  19:Exterior_Entalpic_1           2764 non-null   int64
 19  20:Exterior_Entalpic_2           2764 non-null   int64
 20  21:Exterior_Entalpic_turbo       2764 non-null   int64
 21  22:Temperature_Exterior_Sensor   2764 non-null   float64
 22  23:Humedad_Exterior_Sensor       2764 non-null   float64
 23  24:Day_Of_Week                   2764 non-null   float64
dtypes: float64(19), int64(3), object(2)
memory usage: 518.4+ KB
```

In [9]:
```
data_read.head(5)
```

Out[9]:

	1:Date	2:Time	3:Temperature_Comedor_Sensor	4:Temperature_Habitacion_Sensor	5:Weather_Temperature	6:CO2_Comedor_Sensor	7:CO2_Habitacion
0	13/03/2012	11:45	18.1875	17.8275	0.0	216.560	
1	13/03/2012	12:00	18.4633	18.1207	6.8	219.947	
2	13/03/2012	12:15	18.7673	18.4367	17.0	219.403	
3	13/03/2012	12:30	19.0727	18.7513	18.0	218.613	
4	13/03/2012	12:45	19.3721	19.0414	20.0	217.714	

5 rows × 24 columns

In [10]:
```
data_read.tail()
```

Out[10]:

	1:Date	2:Time	3:Temperature_Comedor_Sensor	4:Temperature_Habitacion_Sensor	5:Weather_Temperature	6:CO2_Comedor_Sensor	7:CO2_Habita
2759	11/04/2012	05:30	21.1520	20.8187	13.0000	190.539	
2760	11/04/2012	05:45	21.0413	20.7053	12.1333	190.421	
2761	11/04/2012	06:00	20.9347	20.5827	12.0000	190.432	
2762	11/04/2012	06:15	20.8560	20.5200	12.0000	191.531	
2763	11/04/2012	06:30	20.7627	20.4400	12.1333	191.563	

5 rows × 24 columns

In [5]:
```
df.describe()
```

Out[5]:

	3:Temperature_Comedor_Sensor	4:Temperature_Habitacion_Sensor	5:Weather_Temperature	6:CO2_Comedor_Sensor	7:CO2_Habitacion_Sensor	8:Humeda
count	2764.000000	2764.000000	2764.000000	2764.000000	2764.000000	
mean	19.199722	18.824852	13.897396	208.479123	211.065844	
std	2.853315	2.821178	4.171991	27.032686	28.469144	
min	11.352000	11.076000	0.000000	187.339000	188.907000	
25%	17.450800	17.060350	10.783325	200.893250	202.682750	
50%	19.373650	19.021000	15.000000	207.045500	209.408000	
75%	21.229975	20.828700	16.666700	211.245500	213.218750	
max	25.540000	24.944000	26.000000	594.389000	609.237000	

8 rows × 22 columns

In [6]:
```
df.columns
```

Out[6]:
```
Index(['1:Date', '2:Time', '3:Temperature_Comedor_Sensor',
       '4:Temperature_Habitacion_Sensor', '5:Weather_Temperature',
       '6:CO2_Comedor_Sensor', '7:CO2_Habitacion_Sensor',
       '8:Humedad_Comedor_Sensor', '9:Humedad_Habitacion_Sensor',
       '10:Lighting_Comedor_Sensor', '11:Lighting_Habitacion_Sensor',
       '12:Precipitacion', '13:Meteo_Exterior_Crepusculo',
       '14:Meteo_Exterior_Viento', '15:Meteo_Exterior_Sol_Oest',
       '16:Meteo_Exterior_Sol_Est', '17:Meteo_Exterior_Sol_Sud',
       '18:Meteo_Exterior_Piranometro', '19:Exterior_Entalpic_1',
       '20:Exterior_Entalpic_2', '21:Exterior_Entalpic_turbo',
       '22:Temperature_Exterior_Sensor', '23:Humedad_Exterior_Sensor',
       '24:Day_Of_Week'],
      dtype='object')
```

```
In [12]:  data_read.dtypes

Out[12]: 1:Date                              object
         2:Time                              object
         3:Temperature_Comedor_Sensor       float64
         4:Temperature_Habitacion_Sensor    float64
         5:Weather_Temperature              float64
         6:CO2_Comedor_Sensor               float64
         7:CO2_Habitacion_Sensor            float64
         8:Humedad_Comedor_Sensor           float64
         9:Humedad_Habitacion_Sensor        float64
         10:Lighting_Comedor_Sensor         float64
         11:Lighting_Habitacion_Sensor      float64
         12:Precipitacion                   float64
         13:Meteo_Exterior_Crepusculo       float64
         14:Meteo_Exterior_Viento           float64
         15:Meteo_Exterior_Sol_Oest         float64
         16:Meteo_Exterior_Sol_Est          float64
         17:Meteo_Exterior_Sol_Sud          float64
         18:Meteo_Exterior_Piranometro      float64
         19:Exterior_Entalpic_1               int64
         20:Exterior_Entalpic_2               int64
         21:Exterior_Entalpic_turbo           int64
         22:Temperature_Exterior_Sensor     float64
         23:Humedad_Exterior_Sensor         float64
         24:Day_Of_Week                     float64
         dtype: object
```

```
In [21]:  data_read.Day_Of_Week.unique()

Out[21]: array([2.     , 2.06667, 3.     , 3.06667, 4.     , 4.06667, 5.     ,
                5.06667, 6.     , 6.06667, 7.     , 6.6    , 1.     , 1.06667])
```

```
In [7]: df = df.drop(['1:Date', '2:Time','19:Exterior_Entalpic_1','20:Exterior_Entalpic_2', '21:Exterior_Entalpic_turbo'], axis = 1)

In [8]: df

Out[8]:
```

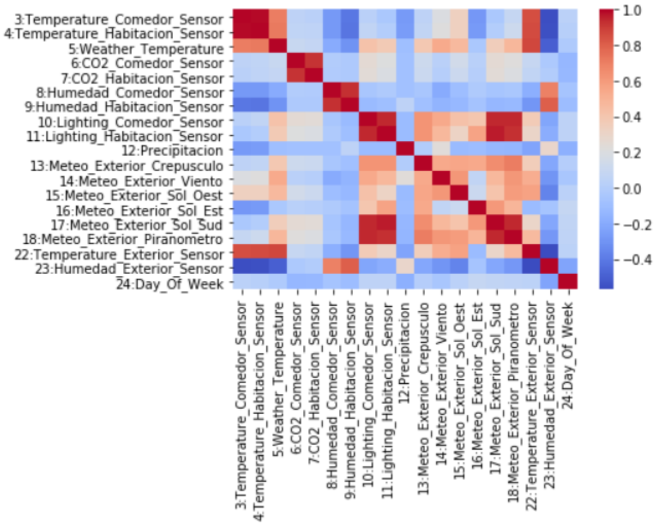
	3:Temperature_Comedor_Sensor	4:Temperature_Habitacion_Sensor	5:Weather_Temperature	6:CO2_Comedor_Sensor	7:CO2_Habitacion_Sensor	8:Humedad
0	18.1875	17.8275	0.0000	216.560	221.920	
1	18.4633	18.1207	6.8000	219.947	220.363	
2	18.7673	18.4367	17.0000	219.403	218.933	
3	19.0727	18.7513	18.0000	218.613	217.045	
4	19.3721	19.0414	20.0000	217.714	216.080	
...	...	...	...	...	...	
2759	21.1520	20.8187	13.0000	190.539	192.181	
2760	21.0413	20.7053	12.1333	190.421	193.067	
2761	20.9347	20.5827	12.0000	190.432	193.653	
2762	20.8560	20.5200	12.0000	191.531	193.387	
2763	20.7627	20.4400	12.1333	191.563	193.664	

2764 rows × 19 columns

3. Plotting and Visualizing Data

Plotting the data with matplotlib scatter

```
In [10]: sns.heatmap(df.corr(),cmap='coolwarm')

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x2183d2cec48>
```



4 Model Fitting, Optimizing, and Predicting

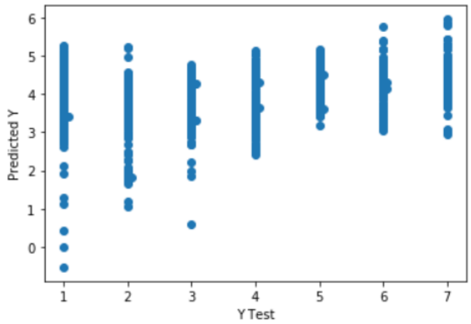
We can now begin to construct our model because our data has been correctly formatted, processed, and organised, and we are aware of the trends and relationships in the general data we are dealing with. We are able to load many classifiers from Sklearn.

```
In [12]: X = df.drop(['24:Day_Of_Week'],axis=1)

In [13]: y = df['24:Day_Of_Week']

In [14]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)

In [15]: lm = LinearRegression()

In [16]: lm.fit(X_train,y_train)

Out[16]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [17]: lm.coef_

Out[17]: array([ 3.93741239e-01, -5.17296327e-01, -9.32903023e-02, -1.77274675e-02,
                 4.91657335e-03,  4.38308777e-02,  2.51336846e-02,  9.35746236e-03,
                 1.72159236e-03,  9.21382605e-02, -3.64554167e-04, -2.30400859e-01,
                 8.86659563e-06, -2.79651819e-06,  3.07732279e-05, -3.64645753e-03,
                 1.10036379e-01, -7.64796004e-02])
```

```
In [18]: predictions = lm.predict(X_test)
```

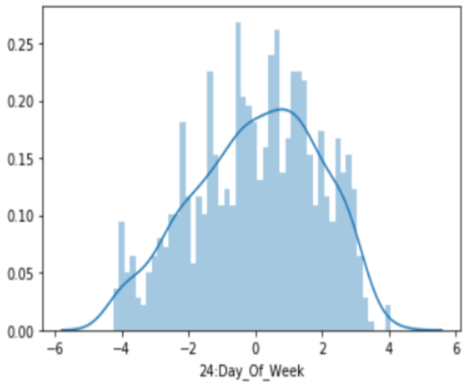
```
In [19]: plt.scatter(y_test,predictions)
         plt.xlabel('Y Test')
         plt.ylabel('Predicted Y')
```

```
Out[19]: Text(0, 0.5, 'Predicted Y')
```




```
In [20]: from sklearn import metrics
         print('MAE: ',metrics.mean_absolute_error(y_test,predictions))
         print('MSE: ',metrics.mean_squared_error(y_test,predictions))
         print('RMSE: ',np.sqrt(metrics.mean_squared_error(y_test,predictions)))

         MAE:  1.5398800021376504
         MSE:  3.435459368440759
         RMSE:  1.8534992226706648
```

```
In [21]: sns.distplot(y_test-predictions,bins=50)

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x21840dfc548>
```



LITERATURE REVIEW

The science of machine learning was expanding quickly in 2010, and new algorithms were being created and used in a variety of applications. The effectiveness, application, and restrictions of various machine learning algorithms in diverse domains, such as classification, regression, and anomaly detection, were being investigated.

Support Vector Machines was a well-liked method at the time (SVMs). Several studies compared the performance of SVMs to other methods, including Random Forest and K-Nearest Neighbors, which were commonly employed for classification applications.

Naive Bayes was a popular technique that was frequently employed for text categorization issues. Naive Bayes classifiers for text classification were thoroughly reviewed by Tax and Duin (2010), who came to the conclusion that they are straightforward, effective, and computationally efficient. They also pointed out that Naive Bayes classifiers can handle high-dimensional data since they are resistant to the dimensionality curse.

In 2010, gradient boosting machines (GBMs) also saw a rise in popularity. Burges et al. (2010) presented the LambdaMART algorithm, a new method for learning a ranking function for information retrieval tasks using gradient descent. On a number of benchmark datasets, they demonstrated that LambdaMART outperformed various cutting-edge algorithms.

The research from 2010 emphasises, in general, the value of choosing the appropriate algorithm for a particular situation as well as the possible advantages of mixing different algorithms to increase prediction accuracy. SVMs, Random Forest, Naive Bayes, and GBMs were all quite popular in 2010, which was a reflection of their efficiency and adaptability to a variety of domains. Building on these principles and continuing to research new algorithms and techniques will be crucial as machine learning develops in order to handle situations that are getting more and more complex.

CONCLUSION

In conclusion, standard machine learning algorithms saw a significant era in 2010 as they gained popularity and were used in a wide variety of applications across numerous sectors. Some of the most widely used algorithms at the time included Support Vector Machines, Random Forest, K-Nearest Neighbors, Naive Bayes, and Gradient Boosting Machines. The relevance of selecting the appropriate algorithm for a specific task and the potential advantages of combining different algorithms to increase prediction accuracy are both highlighted in research articles from this era. Building on the foundations laid in 2010 and continuing to investigate new methodologies are crucial as machine learning develops and new algorithms are created in order to handle more challenging tasks.

REFERENCES

[1]. A Comparison of Random Forest and Support Vector Machines for Land Use Classification, by Jingwei Song, Andrew Hansen, and Zhiliang Zhu.

[2]. An Empirical Comparison of Supervised Learning Algorithms, by Rich Caruana and Alexandru Niculescu-Mizil.

[3]. A Comparative Study of Decision Tree, Random Forest and Boosting Algorithms for Intrusion Detection, by Radhika M. Pai and S. K. Shrivastava.

[4]. Text Classification using Naive Bayes, by Niek Tax and Maarten Duin.

[5]. Learning to Rank with Gradient Descent, by Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender.