

**Modern Education Society's
College of Engineering, Pune-01**

NAME OF STUDENT:	CLASS:
SEMESTER/YEAR:	ROLL NO:
DATE OF PERFORMANCE:	DATE OF SUBMISSION:
EXAMINED BY:	EXPERIMENT NO:

TITLE: ASSIGNMENT ON UBER FARE PREDICTION.

Problem Statement:

Predict the price of the Uber ride from a given pickup point to the agreed drop-off location. Perform following tasks:

1. Pre-process the dataset.
2. Identify outliers.
3. Check the correlation
4. Implement linear regression and random forest regression model.
5. Evaluate the models and compare their respective scores like R2, RMSE etc.

Dataset link: <https://www.kaggle.com/datasets/yasserh/uber-fares-dataset>

Objectives:

1. Understand the Dataset & cleanup (if required).
2. Build Regression models to predict the fare price of uber ride.
3. Also evaluate the models & compare their respective scores like R2, RMSE, etc.

Pre-requisites:

1. Knowledge of python programming.
2. Knowledge of Data Pre-processing.
3. Knowledge of Linear Regression model.
4. Knowledge of the Random Forest Regression model.

Description:

The project is about on world's largest taxi company Uber inc. In this project, we're looking to predict the fare for their future transactional cases. Uber delivers service to lakhs of customers daily. Now it becomes really important to manage their data properly to come up with new business ideas to get best results. Eventually, it becomes really important to estimate the fare prices accurately.

The dataset contains the following fields:

key - a unique identifier for each trip

fare_amount - the cost of each trip in usd

pickup_datetime - date and time when the meter was engaged

passenger_count - the number of passengers in the vehicle (driver entered value)

pickup_longitude - the longitude where the meter was engaged

pickup_latitude - the latitude where the meter was engaged

dropoff_longitude - the longitude where the meter was disengaged

dropoff_latitude - the latitude where the meter was disengaged

Data Pre-processing in Machine learning

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. **It is a data**

mining technique that involves transforming raw **data** into an understandable format. Real-world **data** is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. **Data pre-processing** is a proven method of resolving such issues.

In the real world, **data** are generally incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate **data**. Noisy: containing errors or outliers. Inconsistent: containing discrepancies in codes or names which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

Steps in Data Pre-processing

- Step 1: Get the dataset
- Step 2: Import the libraries
- Step 3: Import the data-set
- Step 4: Check out the missing values
- Step 5: See the Categorical Values
- Step 6: Splitting the data-set into Training and Test Set
- Step 7: Feature Scaling

Step 1: Get the dataset

The collected data for a particular problem in a proper format is known as the **dataset**. Dataset may be of different formats for different purposes, such as, if we want to create a machine learning model for business purpose, then dataset will be different with the dataset required for a liver patient. So each dataset is different from another dataset. To use the dataset in our code, we usually put it into a CSV file. However, sometimes, we may also need to use an HTML or xlsx file.

CSV stands for "**Comma-Separated Values**" files; it is a file format which allows us to save the tabular data, such as spreadsheets. It is useful for huge datasets and can use these datasets in programs. For real-world problems, we can download datasets online from various sources such as:

<https://www.kaggle.com/uciml/datasets>

<https://archive.ics.uci.edu/ml/index.php>

Step 2: Import the Libraries

```
1 # importing libraries
2 import numpy as nm
3 import matplotlib.pyplot as mtp
4 import pandas as pd
5
```

This is how we import libraries in Python using import keyword and this is the most popular libraries which any Data Scientist used. **NumPy** is the fundamental package for scientific computing with Python. It contains among other things:

1. A powerful N-dimensional array object
2. Sophisticated (broadcasting) functions
3. Tools for integrating C/C++ and FORTRAN code
4. Useful linear algebra, Fourier transforms, and random number capabilities

Pandas is for data manipulation and analysis. Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hard copy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

Seaborn is a Python data visualization library based on matplotlib.

It provides a high-level interface for drawing attractive and informative statistical graphics.

Warning messages are typically issued in situations where it is useful to alert the user of some condition in a program, where that condition (normally) doesn't warrant raising an exception and terminating the program. For example, one might want to issue a warning when a program uses an obsolete module.

Step 3: Import the data-set

Now to import the dataset, we will use `read_csv()` function of pandas library, which is used to read a csv file and performs various operations on it. Using this function, we can read a csv file locally as well as through an URL. We can use `read_csv` function as below:

```
Data_set= pd.read_csv('uber.csv')
```

Here, **data_set** is a name of the variable to store our dataset, and inside the function, we have passed the name of our dataset.

Extracting dependent and independent variables:

In machine learning, it is important to distinguish the matrix of features (independent variables) and dependent variables from dataset.

Extracting independent variable:

To extract an independent variable, we will use `iloc[]` method of Pandas library. It is used to extract the required rows and columns from the dataset.

```
x= data_set.iloc[:, :-1].values
```

In the above code, the first colon(:) is used to take all the rows, and the second colon(:) is for all the columns. Here we have used `:-1`, because we don't want to take the last column as it contains the dependent variable. So by doing this, we will get the matrix of features.

Extracting dependent variable:

To extract dependent variables, again, we will use Pandas `.iloc[]` method.

```
y= data_set.iloc[:,3].values
```

Here we have taken all the rows with the last column only. It will give the array of dependent variables.

Step 4: Check out the missing values

The next step of data pre-processing is to handle missing data in the datasets. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset.

Ways to handle missing data:

There are mainly two ways to handle missing data, which are:

By deleting the particular row: The first way is used to commonly deal with null values. In this way, we just delete the specific row or column which consists of null values. But this way is not so efficient and removing data may lead to loss of information which will not give the accurate output.

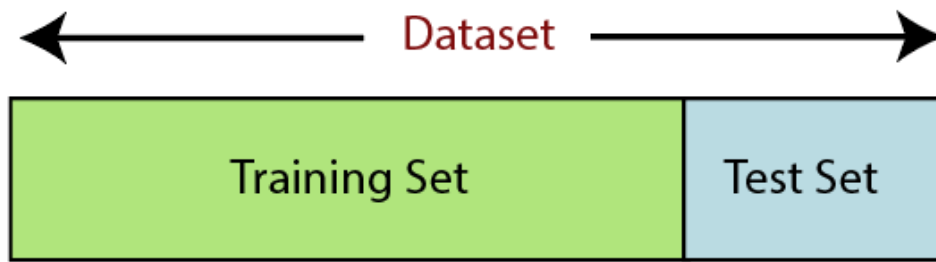
By calculating the mean: In this way, we will calculate the mean of that column or row which contains any missing value and will put it on the place of missing value. This strategy is useful for the features which have numeric data such as age, salary, year, etc.

Step 5: See the Categorical Values

Since machine learning model completely works on mathematics and numbers, but if our dataset would have a categorical variable, then it may create trouble while building the model. So it is necessary to encode these categorical variables into numbers.

Step 6: Splitting the data-set into Training and Test Set

In machine learning data pre-processing, we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model. Here, we can define these datasets as:



Training Set: A subset of dataset to train the machine learning model, and we already know the output.

Test set: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

For splitting the dataset, we will use the below lines of code:

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2, random_state=0)
```

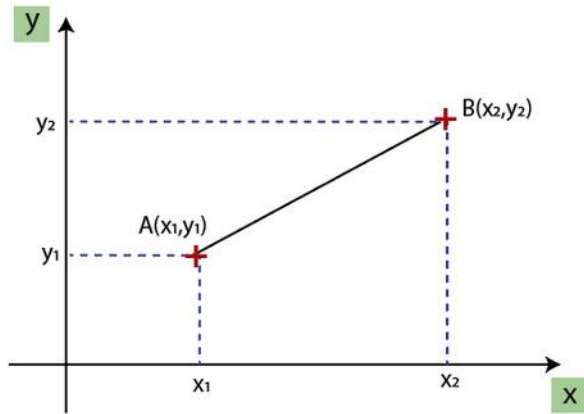
Explanation:

- In the above code, the first line is used for splitting arrays of the dataset into random train and test subsets.
- In the second line, we have used four variables for our output that are
 - **x_train:** features for the training data
 - **x_test:** features for testing data
 - **y_train:** Dependent variables for training data
 - **y_test:** Independent variable for testing data
- In **train_test_split() function**, we have passed four parameters in which first two are for arrays of data, and **test_size** is for specifying the size of the test set. The test_size maybe .5, .3, or .2, which tells the dividing ratio of training and testing sets.
- The last parameter **random_state** is used to set a seed for a random generator so that you always get the same result, and the most used value for this is 42.

Step 7: Feature Scaling

Feature scaling is the final step of data pre-processing in machine learning. It is a technique to standardize the independent variables of the dataset in a specific range. In feature scaling, we put our variables in the same range and in the same scale so that no any variable dominate the other variable. As we can see, the age and salary column values are not on the same scale. A machine learning model is based on **Euclidean distance**, and if we do not scale the variable, then it will

cause some issue in our machine learning model. Euclidean distance is given as:



$$\text{Euclidean Distance Between A and B} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

If we compute any two values from age and salary, then salary values will dominate the age values, and it will produce an incorrect result. So to remove this issue, we need to perform feature scaling for machine learning.

There are two ways to perform feature scaling in machine learning:

Standardization

$$\text{new value } X' = \frac{\text{original value } x - \text{mean}(x)}{a}$$

← mean
← Standard deviation

Normalization

$$\text{new value } X' = \frac{\text{original value } x - \min(x)}{\max(x) - \min(x)}$$

Here, we will use the standardization method for our dataset.

For feature scaling, we will import **StandardScaler** class of **sklearn.preprocessing** library as:

from sklearn.preprocessing import StandardScaler

Now, we will create the object of **StandardScaler** class for independent variables or features. And then we will fit and transform the training dataset.

```
st_x= StandardScaler()
```

```
x_train= st_x.fit_transform(x_train)
```

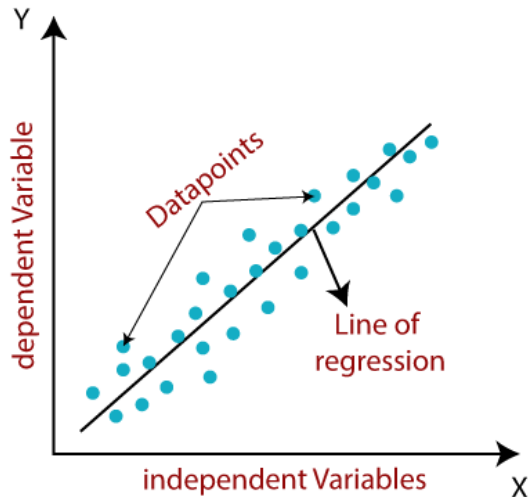
For test dataset, we will directly apply **transform()** function instead of **fit_transform()** because it is already done in training set.

```
x_test= st_x.transform(x_test)
```

Linear Regression in Machine Learning

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price**, etc.

Linear regression is a regression technique in which the independent variable has a linear relationship with the dependent variable. The straight line in the diagram is the best fit line. The main goal of the linear regression is to consider the given data points and plot the best fit line to fit the model in the best way possible.



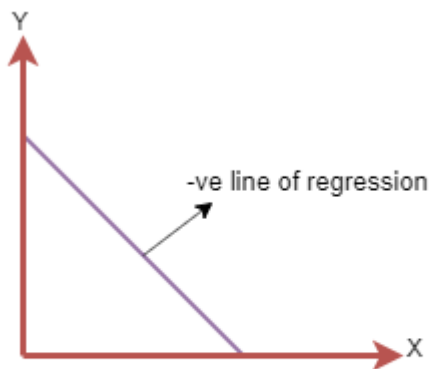
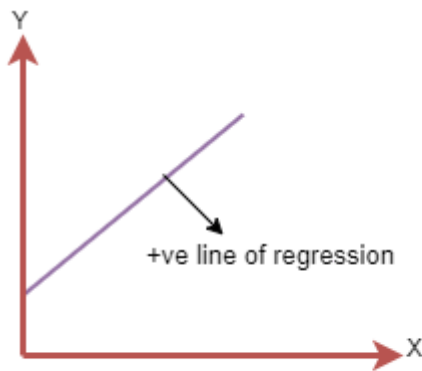
Types of Linear Regression

- **Simple Linear Regression:** If a single independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Simple Linear Regression.
- **Multiple Linear regressions:** If more than one independent variable is used to predict the value of a numerical dependent variable, then such a Linear Regression algorithm is called Multiple Linear Regression.

Linear Regression Line

A linear line showing the relationship between the dependent and independent variables is called a **regression line**. A regression line can show two types of relationship:

- **Positive Linear Relationship:** If the dependent variable increases on the Y-axis and independent variable increases on X-axis, then such a relationship is termed as a Positive linear relationship.



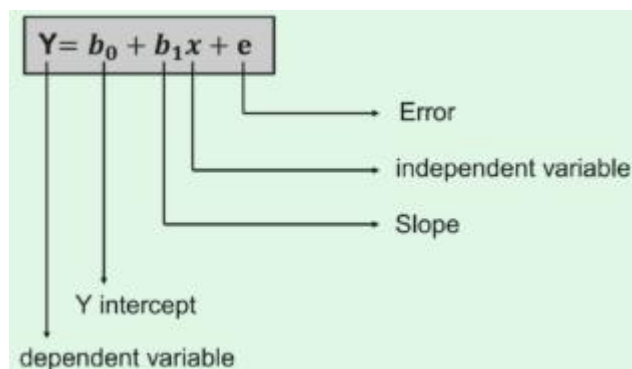
The line of equation will be $Y = b_0 + b_1X$

The line of equation will be $Y = -b_0 + b_1X$

- **Negative Linear Relationship:** If the dependent variable decreases on the Y-axis and independent variable increases on the X-axis, then such a relationship is called a negative linear relationship.

Cost Function

The best fit line can be based on the linear equation given below.



- The dependent variable that is to be predicted is denoted by Y.
- A line that touches the y-axis is denoted by the intercept b_0 .
- b_1 is the slope of the line, x represents the independent variables that determine the prediction of Y.
- The error in the resultant prediction is denoted by e.

The cost function provides the best possible values for b_0 and b_1 to make the best fit line for the data points. The error is minimized between the actual value and the predicted value.

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

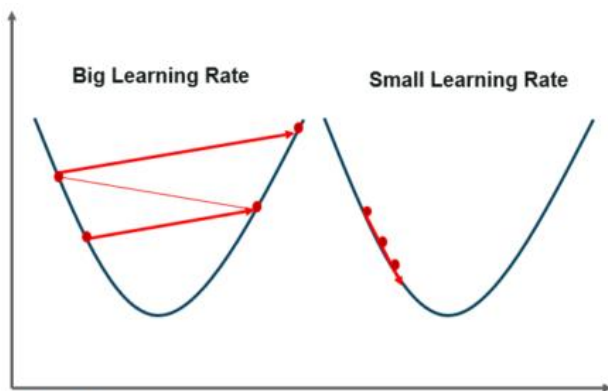
$$J = \frac{1}{n} \sum_{i=1}^n (\text{pred}_i - y_i)^2$$

Above function is used to minimize the error. We square the error difference and sum the error over all data points, the division between the total number of data points. Then, the produced value provides the averaged square error over all data points. It is also known as MSE(Mean Squared Error), and we change the values of b_0 and b_1 so that the MSE value is settled at the minimum.

Gradient Descent

It is a method of updating b_0 and b_1 values to reduce the MSE. The idea behind this is to keep iterating the b_0 and b_1 values until we reduce the MSE to the minimum.

To update b_0 and b_1 , we take gradients from the cost function. To find these gradients, we take partial derivatives with respect to b_0 and b_1 . These partial derivatives are the gradients and are used to update the values of b_0 and b_1 .



A smaller learning rate takes closer to the minimum, but it takes more time and in case of a larger learning rate. The time taken is sooner but there is a chance to overshoot the minimum value.

To implement Linear Regression, the process takes place in the following steps:

1. Loading the Data
2. Exploring the Data
3. Slicing The Data
4. Train and Split Data
5. Generate The Model
6. Evaluate The accuracy

Random Forest Regression in machine learning

Random forest is a supervised learning algorithm that uses an ensemble learning method for classification and regression. Random forest is a bagging technique and not a boosting technique. The trees in random forests run in parallel, meaning is no interaction between these trees while building the trees. Random forest operates by constructing a multitude of decision trees at training

time and outputting the class that's the mode of the classes (classification) or mean prediction (regression) of the individual trees.

The diagram below shows the structure of a Random Forest. You can notice that the trees run in parallel with no interaction amongst them. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees.

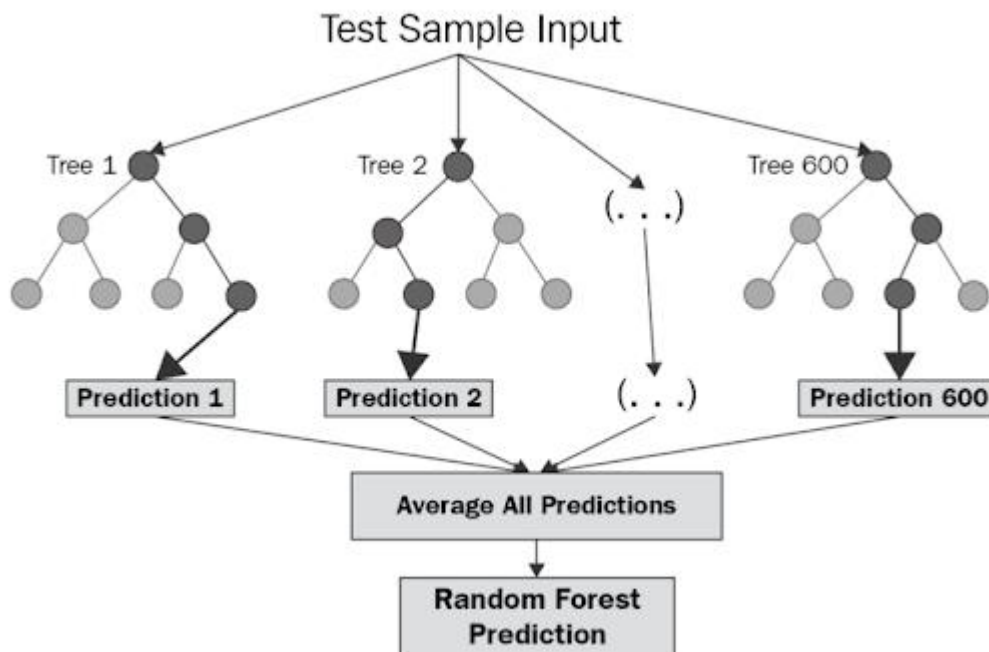


Fig. Random Forest

Implementing Random Forest Regression

1. Importing python libraries and loading our data set into a data frame

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('Position_Salaries.csv')
```

2. Splitting our data set into training set and test set

This step is only for illustrative purposes. There's no need to split this particular data set since we only have 10 values in it.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

3. Creating a random forest regression model and fitting it to the training data

For this model I've chosen 10 trees (**`n_estimators=10`**).

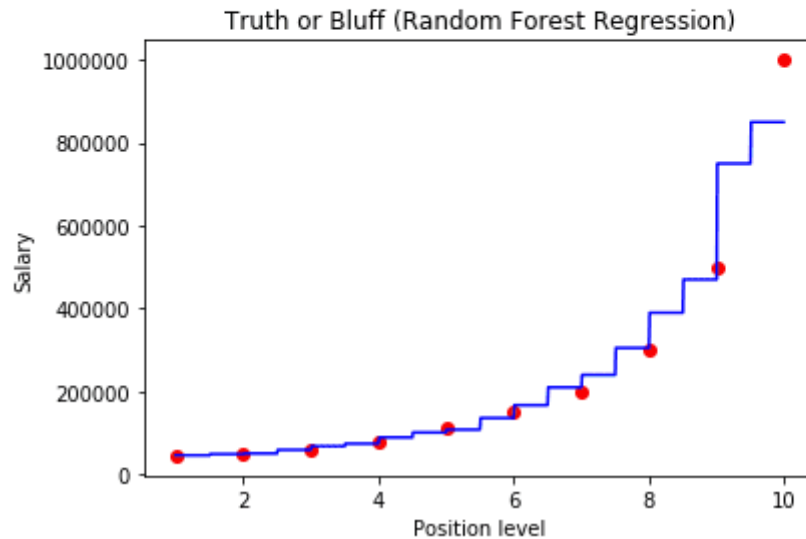
```
# Fitting Random Forest Regression to the dataset
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(X.reshape(-1,1), y.reshape(-1, 1))
```

4. Visualizing the random forest regression results

```

X_grid = np.arange(min(X), max(X), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.title('Truth or Bluff (Random Forest Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()

```



Questions:

1. What are application of linear regression?
2. What are Important Function Used for Linear Regression while program implementation, and explain their purpose?
3. How does a Random Forest Work? What are the advantages of the random forest methodology?