EXPERIMENT . 8

IMPLEMENTATION OF LEARNING ALGORITHMS FOR AN APPLICATION

AIM

To implement learning algorithms for an application

ALGORITHM

- 1. Linear regression is Anding the best equation of line.
- 2. Independent and dependent variables are extracted
- 3. Split the date into training and test datasets. We impost main and test from sklean package
- 4. The next step is to fit the simple linear regression to braining data set.
- 5. The next step is to predict the test results using the linear regression prediction package.
- 6. We then visualise the Atraining set results by plotting the salary training set as graph. Salary is taken on the

Y-axis and years of experience as the X-axis.

I The next step is to find the sessional accuracy lower the value higher will be the ampropy

8 (alculate the errors including mean absolute error.

Mean squared error and root mean squared error.

g Stop

FORMULA

Mean absolute error (MAE) - Distance between any data point and best fit lines

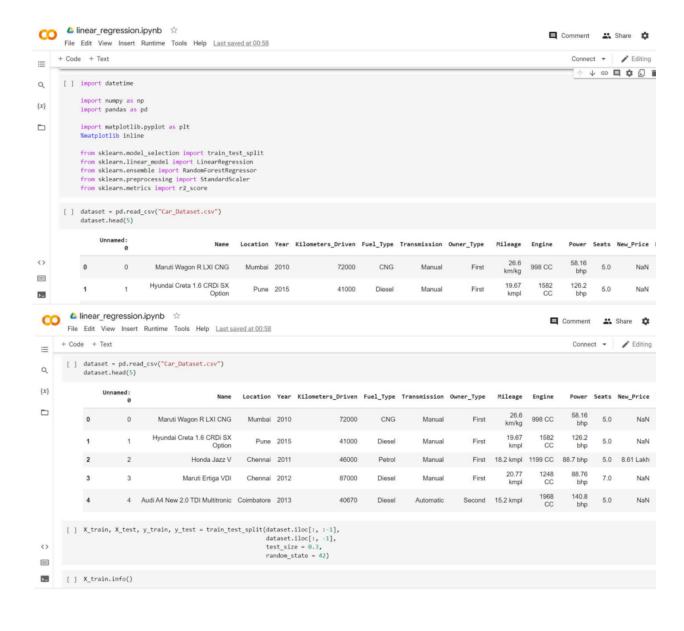
Mean squared error (MSE) - Summation of square of all distances between any data point

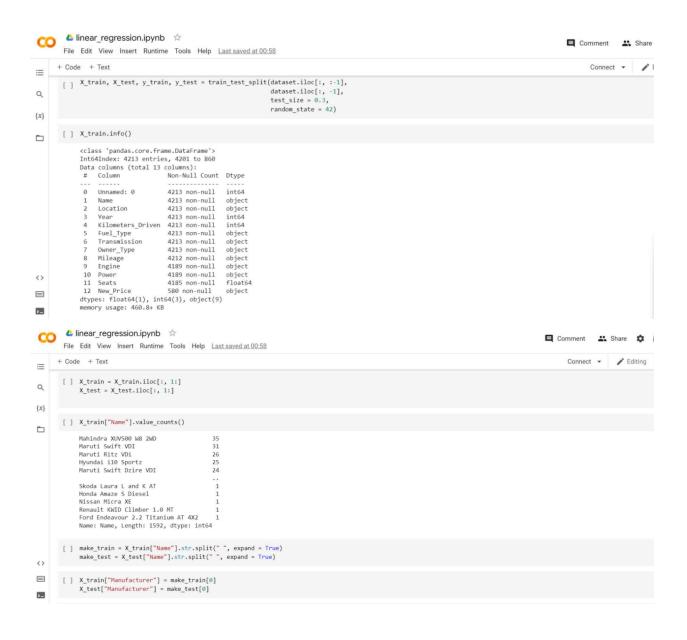
Root Mean Squared Error (RMSE) - Square q MSE $MAE = \frac{1}{n} \frac{S}{i=1} | Ni N |$

MSE = + & (yi - yi)2

RESULT

Learning algorithm for salary dataset was implemented





```
linear_regression.ipynb 
                                                                                                                                                                                                                                                            Comment 👪 Share 🌣
             File Edit View Insert Runtime Tools Help Last saved at 00:58
                                                                                                                                                                                                                                                                       Connect - Editing
=
           [ ] X_train.drop("Name", axis = 1, inplace = True)
X_test.drop("Name", axis = 1, inplace = True)
Q
\{x\}
           [ ] X_train.drop("Location", axis = 1, inplace = True)
X_test.drop("Location", axis = 1, inplace = True)
[ ] curr_time = datetime.datetime.now()
    X_train['Year'] = X_train['Year'].apply(lambda x : curr_time.year - x)
    X_test['Year'] = X_test['Year'].apply(lambda x : curr_time.year - x)
            [ ] X_train["Kilometers_Driven"]
                                   77000
19947
                     4383
                     1779
                                    70963
                                 115195
58752
                                  27000
                     3772
                    5191
5226
                                9000
<>
                    5390
                                   76414
98000
\equiv
                     860
                     Name: Kilometers_Driven, Length: 4213, dtype: int64
>_
            linear_regression.ipynb
CO
                                                                                                                                                                                                                                                           Comment 👪 Share 🌣
            File Edit View Insert Runtime Tools Help Last saved at 00:58
                                                                                                                                                                                                                                                                      Connect - Editing
\equiv
           [ ] mileage_train = X_train["Mileage"].str.split(" ", expand = True)
mileage_test = X_test["Mileage"].str.split(" ", expand = True)
Q
                    \begin{split} &X\_train["Mileage"] = pd.to\_numeric(mileage\_train[\theta], errors = 'coerce') \\ &X\_test["Mileage"] = pd.to\_numeric(mileage\_test[\theta], errors = 'coerce') \end{split} 
{x}
[ ] print(sum(X_train["Mileage"].isnull()))
    print(sum(X_test["Mileage"].isnull()))
          [ ] X_train["Mileage"].fillna(X_train["Mileage"].astype("float64").mean(), inplace = True)
X_test["Mileage"].fillna(X_train["Mileage"].astype("float64").mean(), inplace = True)
           [] cc_train = X_train["Engine"].str.split(" ", expand = True)
cc_test = X_test["Engine"].str.split(" ", expand = True)
X_train["Engine"] = pd.to_numeric(cc_train[0], errors = 'coerce')
X_test["Engine"] = pd.to_numeric(cc_test[0], errors = 'coerce')
<>
                    bhp_train = X_train["Power"].str.split(" ", expand = True)
bhp_test = X_test["Power"].str.split(" ", expand = True)
X_train["Power"] = pd.to_numeric(bhp_train[0], errors = 'coerce')
X_test["Power"] = pd.to_numeric(bhp_test[0], errors = 'coerce')
\equiv
>_
```

