EXPERIMENT - 8

IMPLEMENTATION OF LEARNING ALGORITHMS FOR AN APPLICATION

AIM

To implement learning algorithms for an application

ALGORITHM

1. Linear regression is finding the best equation of line.
2. Independent and dependent variables are extracted
3. Split the date into training and test datasets. We import main and test from sklear package
4. The next step is to fit the simple linear regression to training data set.
5. The next step is to predict the test results using the linear regression prediction package.
6. We then visualise the training 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.

7 The next step is to find the residual accuracy lower the value higher will be the accuracy

8 Calculate the errors including mean absolute Error, Mean squared error and root mean squared error.

9 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 of MSE

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

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

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{N}}$$

RESULT

Learning algorithm for salary dataset was implemented

linear_regression.ipynb

File Edit View Insert Runtime Tools Help Last saved at 00:58

Comment Share

+ Code + Text

Connect Editing

```
[ ] import datetime

    import numpy as np
    import pandas as pd

    import matplotlib.pyplot as plt
    %matplotlib inline

    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LinearRegression
    from sklearn.ensemble import RandomForestRegressor
    from sklearn.preprocessing import StandardScaler
    from sklearn.metrics import r2_score
```

```
[ ] dataset = pd.read_csv("Car_Dataset.csv")
    dataset.head(5)
```

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp	5.0	NaN
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67 kmpl	1582 CC	126.2 bhp	5.0	NaN

linear_regression.ipynb

File Edit View Insert Runtime Tools Help Last saved at 00:58

Comment Share

+ Code + Text

Connect Editing

```
[ ] dataset = pd.read_csv("Car_Dataset.csv")
    dataset.head(5)
```

	Unnamed: 0	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp	5.0	NaN
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67 kmpl	1582 CC	126.2 bhp	5.0	NaN
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18.2 kmpl	1199 CC	88.7 bhp	5.0	8.61 Lakh
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77 kmpl	1248 CC	88.76 bhp	7.0	NaN
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15.2 kmpl	1968 CC	140.8 bhp	5.0	NaN

```
[ ] X_train, X_test, y_train, y_test = train_test_split(dataset.iloc[:, :-1],
                                                        dataset.iloc[:, -1],
                                                        test_size = 0.3,
                                                        random_state = 42)
```

```
[ ] X_train.info()
```

linear_regression.ipynb

File Edit View Insert Runtime Tools Help Last saved at 00:58

Comment Share

+ Code + Text

Connect

```
[ ] X_train, X_test, y_train, y_test = train_test_split(dataset.iloc[:, :-1],
                                                        dataset.iloc[:, -1],
                                                        test_size = 0.3,
                                                        random_state = 42)
```

```
[ ] X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4213 entries, 4201 to 860
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Unnamed: 0         4213 non-null   int64  
 1   Name               4213 non-null   object 
 2   Location           4213 non-null   object 
 3   Year               4213 non-null   int64  
 4   Kilometers_Driven  4213 non-null   int64  
 5   Fuel_Type          4213 non-null   object 
 6   Transmission       4213 non-null   object 
 7   Owner_Type         4213 non-null   object 
 8   Mileage            4212 non-null   object 
 9   Engine             4189 non-null   object 
 10  Power              4189 non-null   object 
 11  Seats              4185 non-null   float64
 12  New_Price          580 non-null    object 
dtypes: float64(1), int64(3), object(9)
memory usage: 460.8+ KB
```

linear_regression.ipynb

File Edit View Insert Runtime Tools Help Last saved at 00:58

Comment Share

+ Code + Text

Connect Editing

```
[ ] X_train = X_train.iloc[:, 1:]
    X_test = X_test.iloc[:, 1:]
```

```
[ ] X_train["Name"].value_counts()
```

```
Mahindra XUV500 W8 2WD               35
Maruti Swift VDI                     31
Maruti Ritz VDi                      26
Hyundai i10 Sportz                   25
Maruti Swift Dzire VDI               24
                                     ..
Skoda Laura L and K AT                1
Honda Amaze S Diesel                  1
Nissan Micra XE                       1
Renault KWID Climber 1.0 MT           1
Ford Endeavour 2.2 Titanium AT 4X2    1
Name: Name, Length: 1592, dtype: int64
```

```
[ ] make_train = X_train["Name"].str.split(" ", expand = True)
    make_test = X_test["Name"].str.split(" ", expand = True)
```

```
[ ] X_train["Manufacturer"] = make_train[0]
    X_test["Manufacturer"] = make_test[0]
```

linear_regression.ipynb

File Edit View Insert Runtime Tools Help Last saved at 00:58

Comment Share

+ Code + Text Connect Editing

```
[ ] X_train.drop("Name", axis = 1, inplace = True)
    X_test.drop("Name", axis = 1, inplace = True)
```

```
[ ] 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"]
```

```
4201     77000
4383     19947
1779     70963
4020    115195
3248     58752
         ...
3772     27000
5191      9000
5226    140000
5390     76414
860      98000
Name: Kilometers_Driven, Length: 4213, dtype: int64
```

linear_regression.ipynb

File Edit View Insert Runtime Tools Help Last saved at 00:58

Comment Share

+ Code + Text Connect Editing

```
[ ] mileage_train = X_train["Mileage"].str.split(" ", expand = True)
    mileage_test = X_test["Mileage"].str.split(" ", expand = True)

    X_train["Mileage"] = pd.to_numeric(mileage_train[0], errors = 'coerce')
    X_test["Mileage"] = pd.to_numeric(mileage_test[0], errors = 'coerce')
```

```
[ ] print(sum(X_train["Mileage"].isnull()))
    print(sum(X_test["Mileage"].isnull()))
```

```
1
1
```

```
[ ] 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')
```

linear_regression.ipynb

File Edit View Insert Runtime Tools Help Last saved at 00:58

Comment Share

+ Code + Text

Connect Editing

```
[ ] X_train["Engine"].fillna(X_train["Engine"].astype("float64").mean(), inplace = True)
    X_test["Engine"].fillna(X_train["Engine"].astype("float64").mean(), inplace = True)

    X_train["Power"].fillna(X_train["Power"].astype("float64").mean(), inplace = True)
    X_test["Power"].fillna(X_train["Power"].astype("float64").mean(), inplace = True)

    X_train["Seats"].fillna(X_train["Seats"].astype("float64").mean(), inplace = True)
    X_test["Seats"].fillna(X_train["Seats"].astype("float64").mean(), inplace = True)
```

```
[ ] X_train.drop(["New_Price"], axis = 1, inplace = True)
    X_test.drop(["New_Price"], axis = 1, inplace = True)
```

```
[ ] X_train = pd.get_dummies(X_train,
                             columns = ["Manufacturer", "Fuel_Type", "Transmission", "Owner_Type"],
                             drop_first = True)
```

```
[ ] X_test = pd.get_dummies(X_test,
                            columns = ["Manufacturer", "Fuel_Type", "Transmission", "Owner_Type"],
                            drop_first = True)
```

```
missing_cols = set(X_train.columns) - set(X_test.columns)
for col in missing_cols:
    X_test[col] = 0
```

linear_regression.ipynb

File Edit View Insert Runtime Tools Help Last saved at 00:58

Comment Share

+ Code + Text

Connect Editing

```
[ ] missing_cols = set(X_train.columns) - set(X_test.columns)
    for col in missing_cols:
        X_test[col] = 0
    X_test = X_test[X_train.columns]
```

```
[ ] standardScaler = StandardScaler()
    standardScaler.fit(X_train)
    X_train = standardScaler.transform(X_train)
    X_test = standardScaler.transform(X_test)
```

```
[ ] linearRegression = LinearRegression()
    linearRegression.fit(X_train, y_train)
    y_pred = linearRegression.predict(X_test)
    r2_score(y_test, y_pred)
```

```
0.7008908549416727
```

```
[ ] rf = RandomForestRegressor(n_estimators = 100)
    rf.fit(X_train, y_train)
    y_pred = rf.predict(X_test)
    r2_score(y_test, y_pred)
```

```
0.8882005989619215
```