Multiple Linear Regression Using Python: Machine Learning

Here we can learn how to build a multiple linear regression model in machine learning using Visual Studio Code.

If we have more than one independent variables then instead of using simple linear regrassion we use multiple linear regression model.

The line equation for multiple linear regression model is:

$$ y = \beta_0 + \beta_1X_1 + \beta_2X_2 + \beta_3X_3 ; \dotsc : \dotsc + \beta_iX_i + e
$$

The dataset contains information about the 50 startups. Features include R&D Spend, Administration, Marketing Spend, State and Profit.

Let's start by importing some libraries

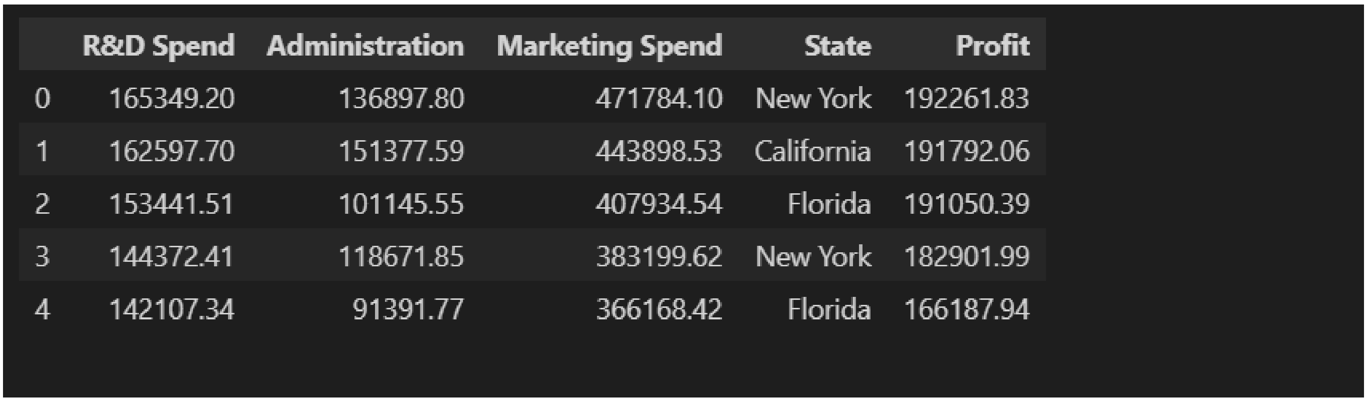
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Import dataset

Import the dataset and view it's first five columns. dot head() will show first five rows of dataset.

```
data_set = pd.read_csv('compList.csv')

data_set.head()
```



	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

View the info of our dataframe, it shows that all the columns are numerical except State column. State has categorical values

```
data_set.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   R&D Spend        50 non-null     float64
 1   Administration   50 non-null     float64
 2   Marketing Spend  50 non-null     float64
 3   State            50 non-null     object 
 4   Profit           50 non-null     float64
dtypes: float64(4), object(1)
memory usage: 2.1+ KB
```

View the shape of the dataset.

```
print("Dataset contains {} rows and {} columns".format(shape[0],shape[1]))
```

```
Dataset contains 50 rows and 5 columns
```

View all the columns in dataset.

```
data_set.columns
```

```
Index(['R&D Spend', 'Administration', 'Marketing Spend', 'State', 'Profit'],
dtype='object')
```

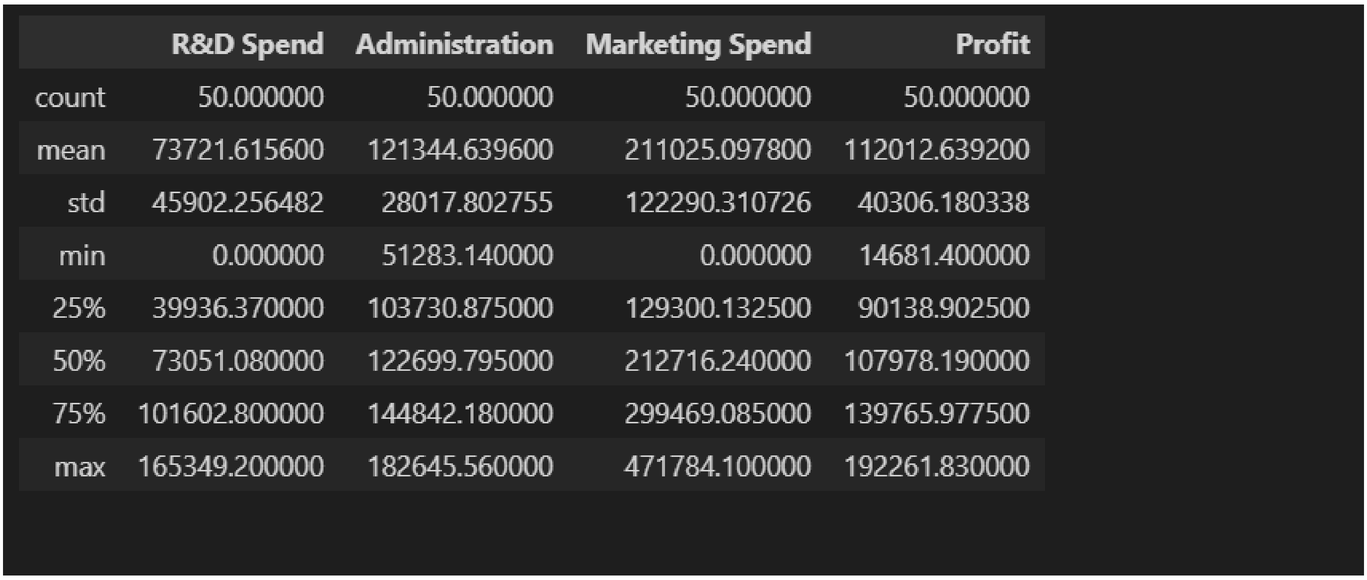
View all the unique values in column "State".

```
data_set['State'].unique()
```

```
array(['New York', 'California', 'Florida'], dtype=object)
```

Statistical Details of the dataset

```
data_set.describe()
```

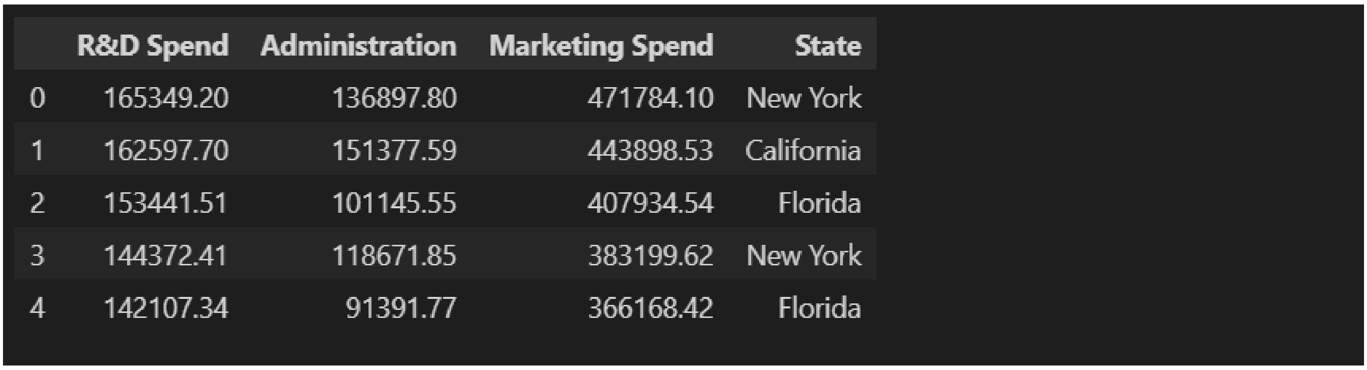


	R&D Spend	Administration	Marketing Spend	Profit
count	50.000000	50.000000	50.000000	50.000000
mean	73721.615600	121344.639600	211025.097800	112012.639200
std	45902.256482	28017.802755	122290.310726	40306.180338
min	0.000000	51283.140000	0.000000	14681.400000
25%	39936.370000	103730.875000	129300.132500	90138.902500
50%	73051.080000	122699.795000	212716.240000	107978.190000
75%	101602.800000	144842.180000	299469.085000	139765.977500
max	165349.200000	182645.560000	471784.100000	192261.830000

Now we will define the Descriptive Features and Target Features Separately

Extract Descriptive Features from dataset

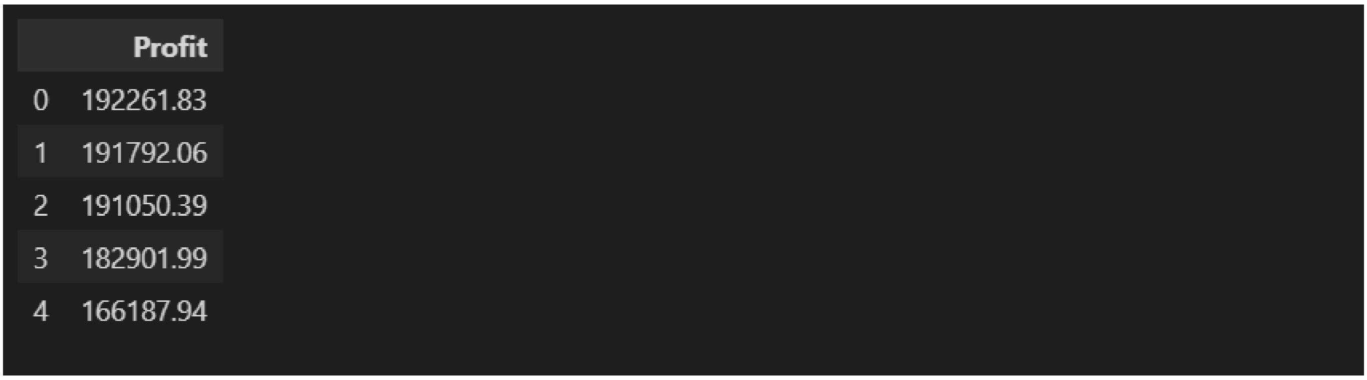
```
X = data_set.iloc[:,:-1]
X.head()
```



	R&D Spend	Administration	Marketing Spend	State
0	165349.20	136897.80	471784.10	New York
1	162597.70	151377.59	443898.53	California
2	153441.51	101145.55	407934.54	Florida
3	144372.41	118671.85	383199.62	New York
4	142107.34	91391.77	366168.42	Florida

Extract Target Features from dataset

```
y = data_set.iloc[:,-1:]
y.head()
```



	Profit
0	192261.83
1	191792.06
2	191050.39
3	182901.99
4	166187.94

Perform One-Hot Encoding

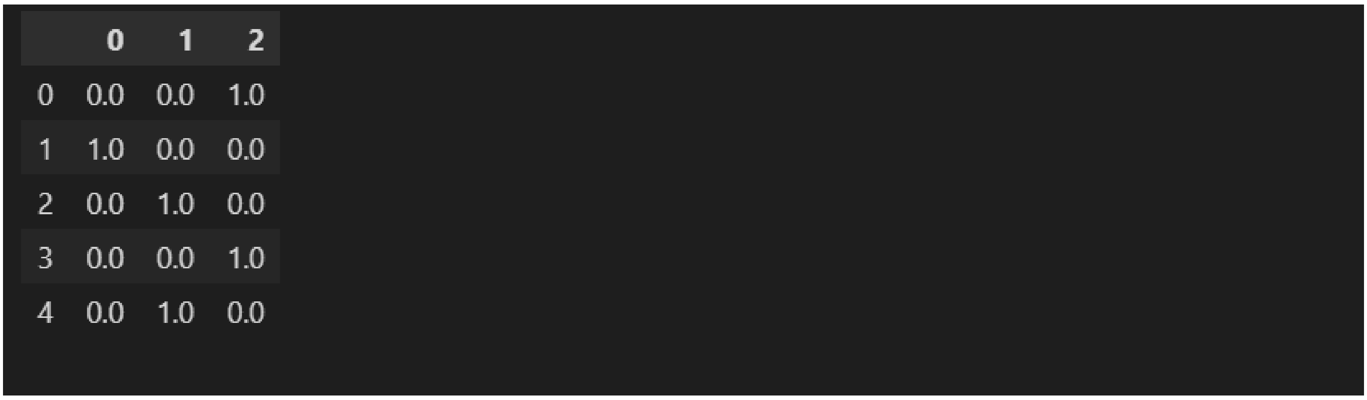
We will use One-Hot Encoding because we have categorical values in the dataset. If we look at the "State" column we have catagorical values in it. So we have to use One-Hot Encoding to convert them into binary combinations for further analysis.

```
# Using OneHoteEncoder

from sklearn.preprocessing import OneHotEncoder
one_hot_encoder = OneHotEncoder(handle_unknown='ignore')

X_encoder = pd.DataFrame(one_hot_encoder.fit_transform(X[['State']]).toarray())

X_encoder.head()
```

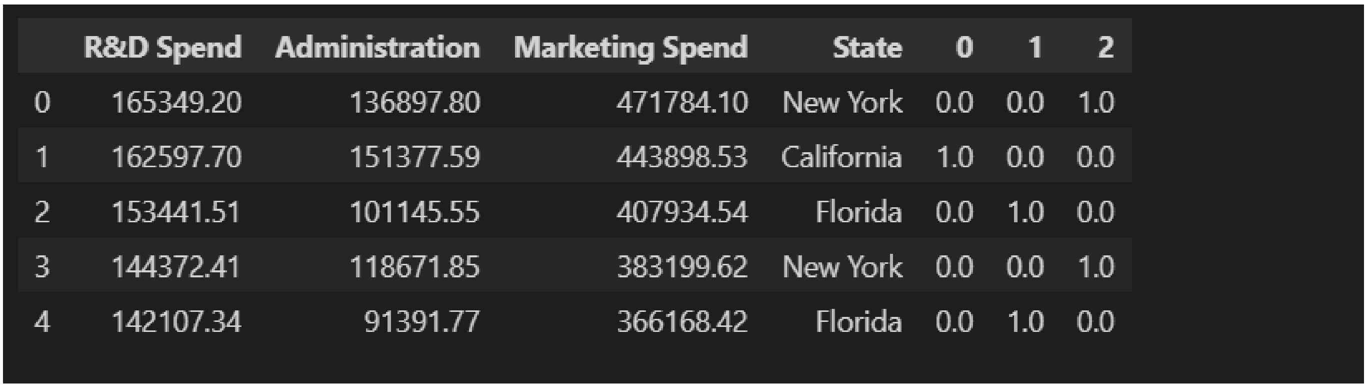


	0	1	2
0	0.0	0.0	1.0
1	1.0	0.0	0.0
2	0.0	1.0	0.0
3	0.0	0.0	1.0
4	0.0	1.0	0.0

Now join X_encoder with the Descriptive Features (X).

```
final_X = X.join(X_encoder)

final_X.head()
```

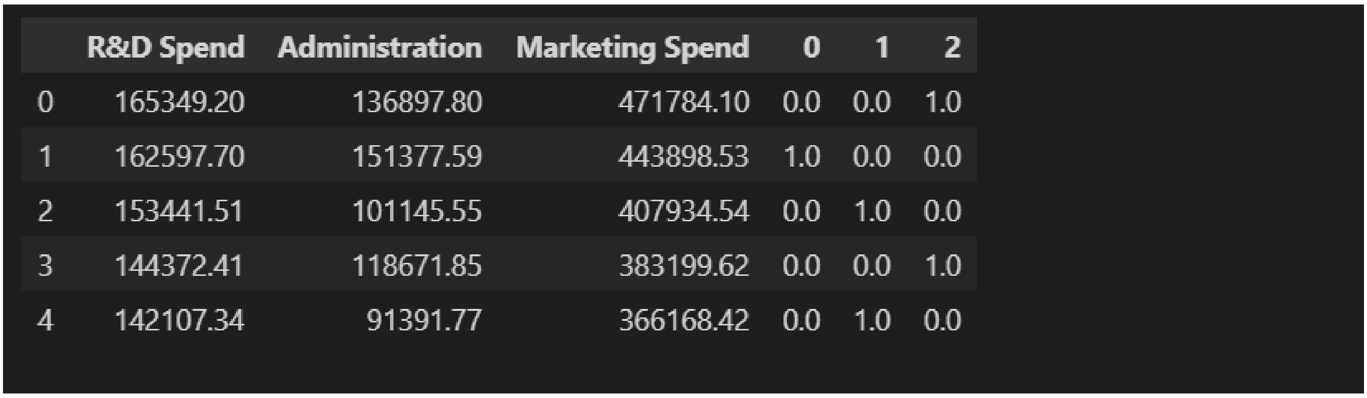


	R&D Spend	Administration	Marketing Spend	State	0	1	2
0	165349.20	136897.80	471784.10	New York	0.0	0.0	1.0
1	162597.70	151377.59	443898.53	California	1.0	0.0	0.0
2	153441.51	101145.55	407934.54	Florida	0.0	1.0	0.0
3	144372.41	118671.85	383199.62	New York	0.0	0.0	1.0
4	142107.34	91391.77	366168.42	Florida	0.0	1.0	0.0

Drop the original categorical variable (State)

```
final_X.drop('State', axis=1, inplace=True)

final_X.head()
```



	R&D Spend	Administration	Marketing Spend	0	1	2
0	165349.20	136897.80	471784.10	0.0	0.0	1.0
1	162597.70	151377.59	443898.53	1.0	0.0	0.0
2	153441.51	101145.55	407934.54	0.0	1.0	0.0
3	144372.41	118671.85	383199.62	0.0	0.0	1.0
4	142107.34	91391.77	366168.42	0.0	1.0	0.0

Splitting the Dataset into Training Set and Test Set

Now split the dataset into two parts i.e., 80% data will go for training set and 20% dataset will go for test set. You can also choose 70-80, 60-40 or 50-50 training and testing set or according to your choice.

```
# Splitting the dataset into training set and test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(final_X, y, test_size=0.2,
random_state=0)
```

Fitting the Multiple Linear Regression model to the training set

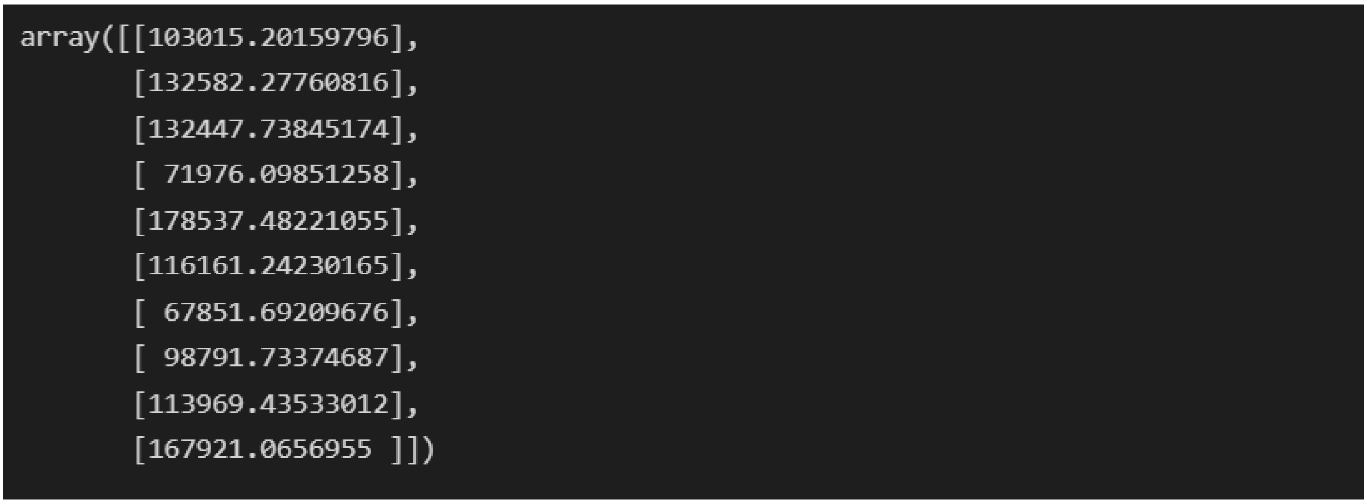
```
#Fitting the Multiple Linear Regression model to the training set

from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```

Predicting the Test set result

```
y_pred = model.predict(X_test)
y_pred
```

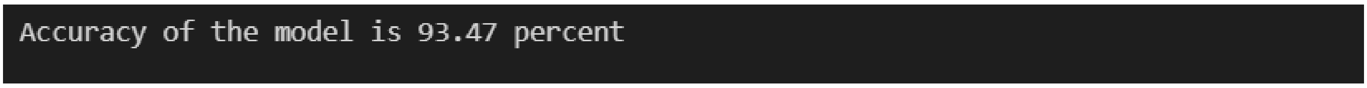


```
array([[103015.20159796],
       [132582.27760816],
       [132447.73845174],
       [ 71976.09851258],
       [178537.48221055],
       [116161.24230165],
       [ 67851.69209676],
       [ 98791.73374687],
       [113969.43533012],
       [167921.0656955 ]])
```

Check Accuracy of the model

```
score =  model.score(X_test, y_test)*100

print('Accuracy of the model is %.2f percent' %score)
```

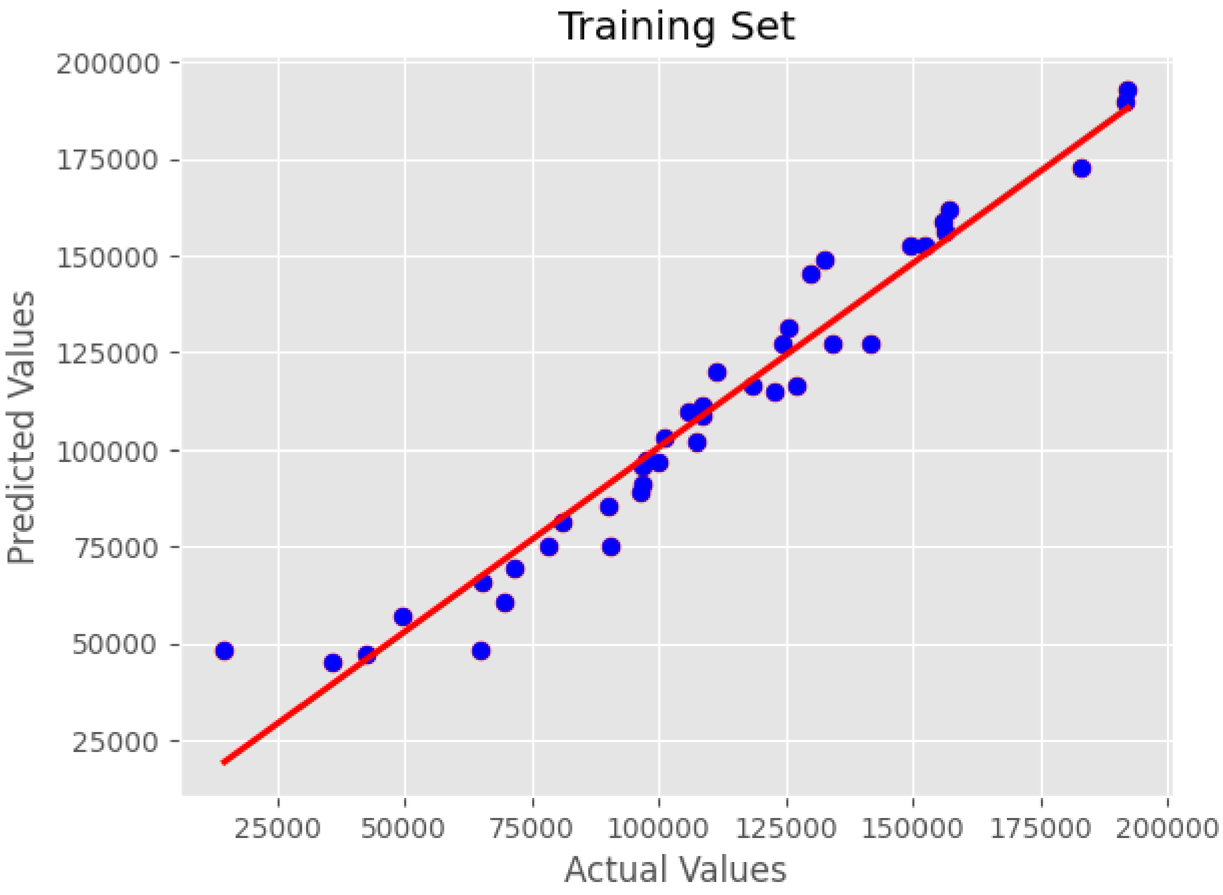


```
Accuracy of the model is 93.47 percent
```

Plot the Results

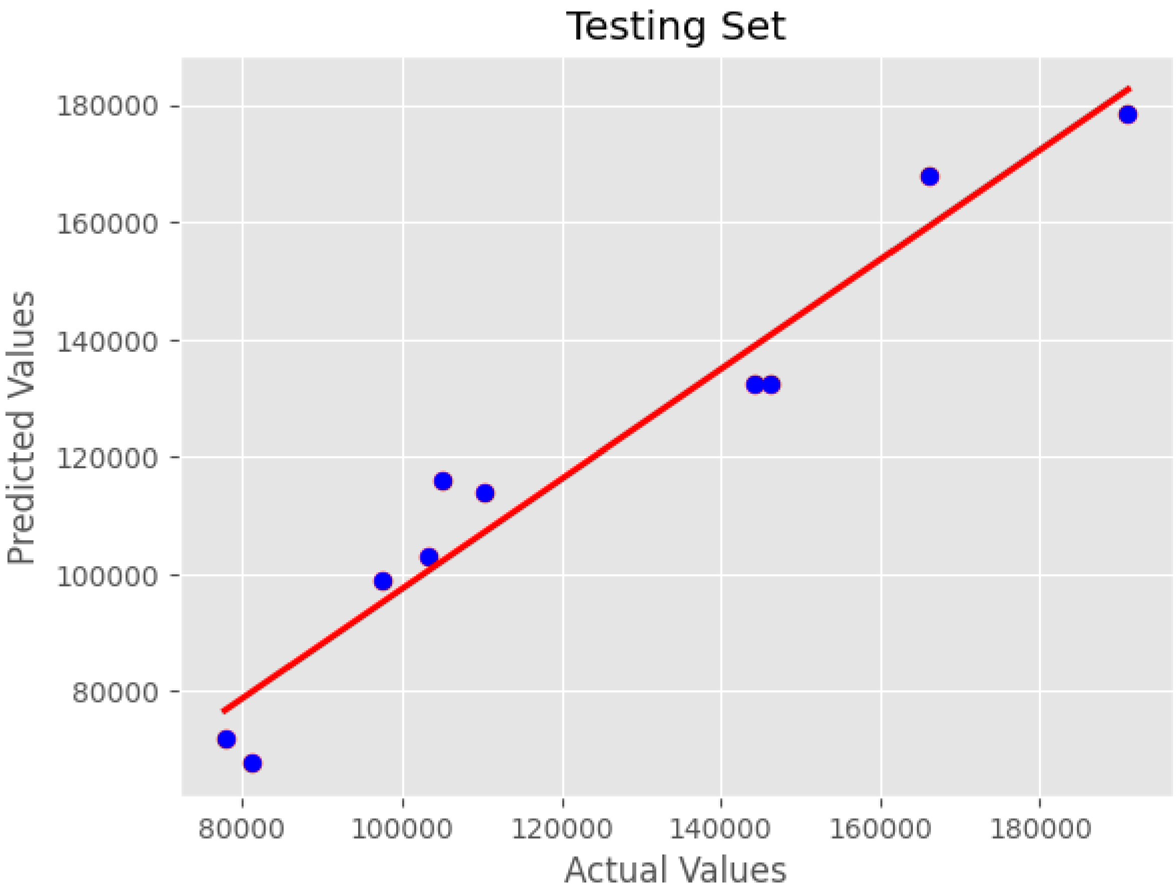
Visualizing the Training set results

```
sns.regplot(x=y_train,y=pred_y,ci=None,color ='red')
plt.scatter(y_train,pred_y, color = 'blue', label = 'sample')
plt.title('Training Set')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
```



Visualizing the Test set results

```
sns.regplot(x=y_test,y=y_pred,ci=None,color ='red')
plt.scatter(y_test,y_pred, color = 'blue', label = 'sample')
plt.title('Testing Set')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
```



Conclusion

We have created a new Multiple Linear Regression Model and we learned how to perform One-Hot Encoding for Categerical Values.