assignment

April 1, 2024

[1]:
```
import pandas as pd
```

[2]:
```
df = df = pd.read_csv('../../Data/automobileEDA.csv')
df.head()
```

[2]:
```
   symboling  normalized-losses         make aspiration num-of-doors  \
0          3                122  alfa-romero        std          two
1          3                122  alfa-romero        std          two
2          1                122  alfa-romero        std          two
3          2                164         audi        std         four
4          2                164         audi        std         four

    body-style drive-wheels engine-location  wheel-base    length  …  \
0  convertible          rwd           front        88.6  0.811148  …
1  convertible          rwd           front        88.6  0.811148  …
2    hatchback          rwd           front        94.5  0.822681  …
3        sedan          fwd           front        99.8  0.848630  …
4        sedan          4wd           front        99.4  0.848630  …

   compression-ratio  horsepower  peak-rpm city-mpg highway-mpg    price  \
0                9.0       111.0    5000.0       21          27  13495.0
1                9.0       111.0    5000.0       21          27  16500.0
2                9.0       154.0    5000.0       19          26  16500.0
3               10.0       102.0    5500.0       24          30  13950.0
4                8.0       115.0    5500.0       18          22  17450.0

  city-L/100km  horsepower-binned  diesel  gas
0    11.190476             Medium       0    1
1    11.190476             Medium       0    1
2    12.368421             Medium       0    1
3     9.791667             Medium       0    1
4    13.055556             Medium       0    1

[5 rows x 29 columns]
```

[5]:
```
x = df.drop("price", axis=1)
y = df['price']
```

[6]:
```python
from sklearn.model_selection import train_test_split
```

[48]:
```python
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
↪random_state=42)
```

[49]:
```python
to_select = ['horsepower','curb-weight','engine-size','highway-mpg']
x_train = X_train[to_select].values
x_test = X_test[to_select].values
```

[3]:
```python
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

[52]:
```python
model.fit(x_train, y_train)
```

[52]:
```
LinearRegression()
```

[58]:
```python
y_pred_train = model.predict(x_train)
y_pred_test = model.predict(x_test)
```

[59]:
```python
model.intercept_
```

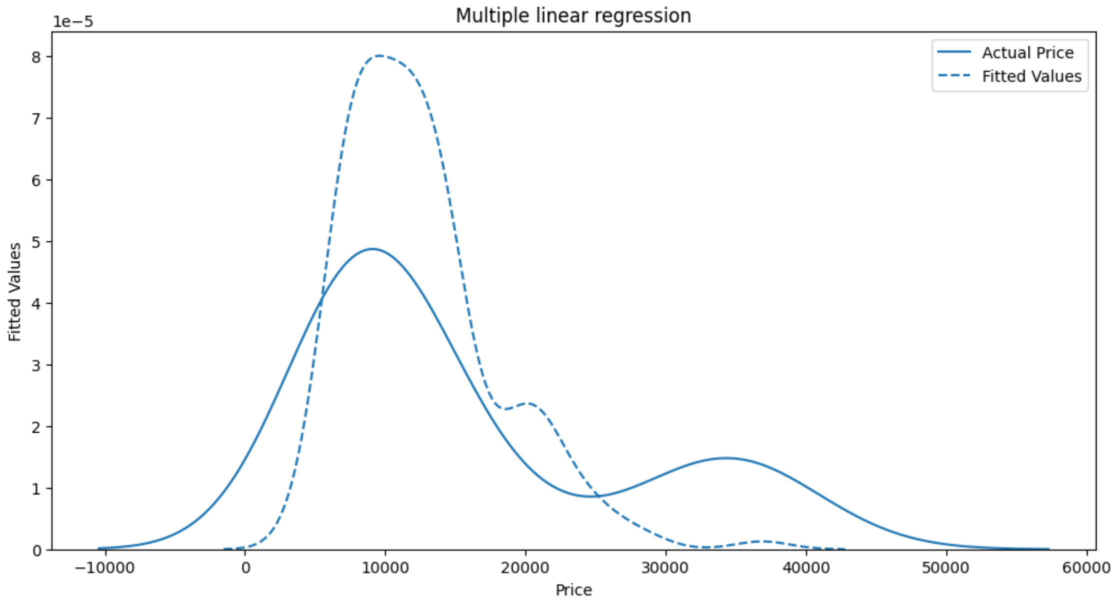
[59]:
```
-8009.241208155732
```

[61]:
```python
model.coef_
```

[61]:
```
array([ 25.9407567 ,   3.7778736 ,  83.14757024, -65.72539722])
```

[4]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
```

[41]:
```python
plt.figure(figsize=(12, 6))
ax1 = sns.kdeplot(y_test, color='r', label='Actual Price')
sns.kdeplot(y_pred_train, color='b', label='Fitted Values', ax=ax1,␣
↪linestyle='--') #Color is not changed so linestyle is changed
plt.title("Multiple linear regression")
plt.xlabel("Price")
plt.ylabel("Fitted Values")
plt.legend()
plt.show()
plt.close()
```



0.1 Mean Square Error (MSE)

The Mean Square Error (MSE) is a common loss function used in regression problems to measure the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. The MSE is always non-negative, and values closer to zero are better. The formula for calculating MSE is given by:

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

Where: - n is the number of observations, - y_i is the actual value of the i-th observation, - $\hat{y}_i$ is the predicted value for the i-th observation.

[7]:
```
x= df['horsepower'].values.reshape(-1, 1)
y = df['price'].values.reshape(-1, 1)

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
↪random_state=42)
model = LinearRegression()
model.fit(X_train, y_train)
```

[7]: LinearRegression()

[8]:
```
y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)
```

[10]:
```
from sklearn.metrics import mean_squared_error
```

[11]:
```
mse_train = mean_squared_error(y_train, y_pred_train)
mse_test = mean_squared_error(y_test, y_pred_test)

print(f"Train Mean Squared Error: {mse_train}")
print(f"Test Mean Squared Error: {mse_test}")

print('\n')
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

```
Train Mean Squared Error: 16320010.468020136
Test Mean Squared Error: 46093655.049259596


Coefficients: [[151.88933994]]
Intercept: [-2907.17676069]
```

0.2 MSE using numpy

[12]:
```
import numpy as np
```

[13]:
```
mse=np.mean((y_train - y_pred_train) ** 2)
mse
```

[13]: 16320010.468020136

0.3 MSE using sklearn

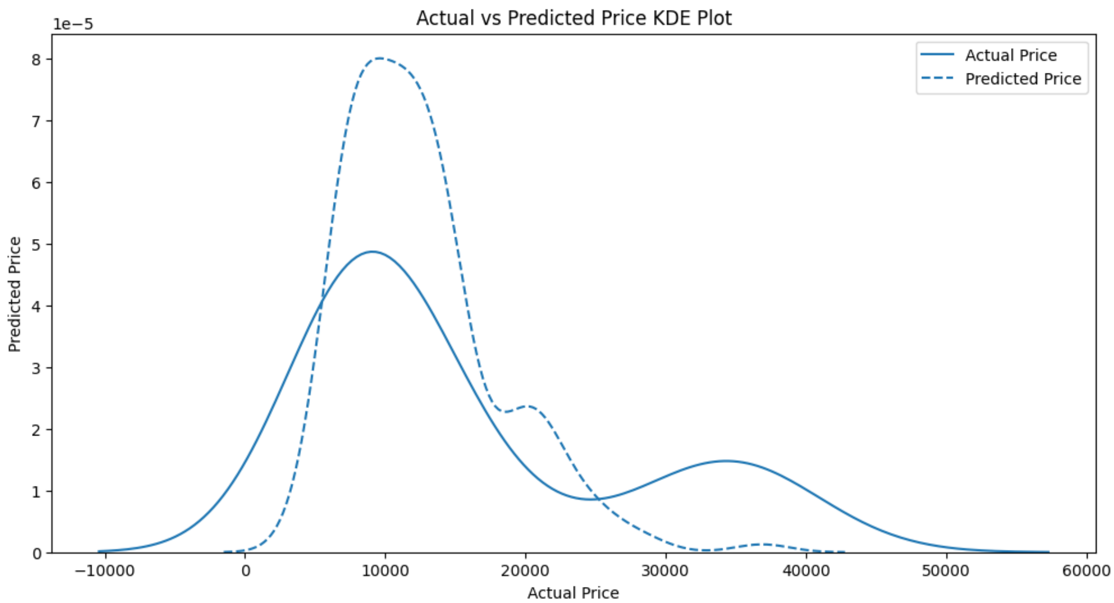
[66]:
```
from sklearn.metrics import mean_squared_error
```

[68]:
```
mse_train = mean_squared_error(y_train, y_pred_train)
mse_test = mean_squared_error(y_test, y_pred_test)
mse_train, mse_test
```

[68]: (8719832.96736771, 28916467.462179784)

0.4 Visualization

[40]:
```
plt.figure(figsize=(12,6))
ax1 = sns.kdeplot(y_test,  label='Actual Price')
sns.kdeplot(y_pred_train,  label= 'Predicted Price',  ax=ax1,␣
 ↪linestyle='--')#Color is not changed so linestyle is changed
plt.xlabel('Actual Price')
plt.ylabel('Predicted Price')
plt.title('Actual vs Predicted Price KDE Plot')
plt.legend()
plt.show()
```



0.5 Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) is a metric used to measure the average magnitude of the errors between pairs of predictions and actual outcomes, without considering their direction. It's the mean over the test sample of the absolute differences between predicted and actual values. MAE provides a straightforward indication of average error magnitude in the same units as the data. The MAE formula is:

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

Where: - n is the number of observations, - y_i is the actual value for the i-th observation, - $\hat{y}_i$ is the predicted value for the i-th observation.

MAE is particularly useful for understanding the average error magnitude directly in the output variable's units.

[15]:
```python
from sklearn.metrics import mean_absolute_error
```

[16]:
```python
mse_train = mean_absolute_error(y_train, y_pred_train)
mse_test = mean_absolute_error(y_test, y_pred_test)
mse_train, mse_test
```

[16]: (2877.7586831038134, 4809.393573026446)

0.6 Root Mean Square Error (RMSE)

The Root Mean Square Error (RMSE) is another metric used to measure the average magnitude of the error. It's the square root of the average of squared differences between the predicted values and actual values. Compared to MAE, RMSE gives a relatively high weight to large errors. This means RMSE is more sensitive to outliers than MAE. The RMSE formula is:

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

Where: - n is the number of observations, - y_i is the actual value of the i-th observation, - $\hat{y}_i$ is the predicted value for the i-th observation.

RMSE is valuable when large errors are particularly undesirable and should be penalized more than smaller errors. It is also more appropriate than MAE when the error distribution is expected to be Gaussian.

[17]:
```python
from sklearn.metrics import root_mean_squared_error
```

[18]:
```python
mse_train = root_mean_squared_error(y_train, y_pred_train)
mse_test = root_mean_squared_error(y_test, y_pred_test)
mse_train, mse_test
```

[18]: (4039.803270955176, 6789.230814257208)