stat_3

February 20, 2023

[ ]:

```python
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
```

[ ]:

```python
# create the dataframe
df = pd.read_csv ('cars_clean.csv')
```

[ ]:

```python
# size of dataframe
df.shape
```

Linear Regression

y = a + b * x

[ ]:

```python
# import linear regression model from scikit-learn
from sklearn.linear_model import LinearRegression
```

[ ]:

```python
# create the linear regression object
lm = LinearRegression()
```

[ ]:

```python
# can highway-mpg predict car price
X = df[['highway-mpg']]
Y = df['price']
```

[ ]:

```python
# fit the linear model
lm.fit (X, Y)
```

[ ]:

```python
# output prediction
Yhat = lm.predict (X)
Yhat[0:5]
```

[ ]:

```python
# what is the intercept
lm.intercept_
```

[ ]:

```
# what is the slope
lm.coef_
```

R^2: Coefficient of Determination

[ ]:

```
print ('R-square = ', lm.score(X,Y))
```

Mean Squared Error

[ ]:

```
# import mean square error module
from sklearn.metrics import mean_squared_error
```

[ ]:

```
# obtain mean squared error
mse = mean_squared_error (df['price'], Yhat)
print ('MSE = ', mse)
```

Multiple Linear Regression

[ ]:

```
# define predictor variables
Z = df[['engine-size', 'highway-mpg']]
```

[ ]:

```
# create the model
lm.fit (Z, df['price'])
```

[ ]:

```
# value of the intercept
lm.intercept_
```

[ ]:

```
# value of the coefficients
lm.coef_
```

price = 2903.80 + 139.84 * engine-size - 242.41 * highway-mpg

Model Evaluation using Visualization

[ ]:

```
# regression plot
sns.regplot (x = 'highway-mpg', y ='price', data = df)
plt.ylim(0,)
plt.show()
```

[ ]:

```
# regression plot
sns.regplot (x = 'engine-size', y ='price', data = df)
plt.ylim(0,)
plt.show()
```

Residual Plot

[ ]:

```
# create a residual plot
sns.residplot (x = df['highway-mpg'], y = df['price'])
plt.show()
```

Distribution Plot

[ ]:
```
Y_hat = lm.predict(Z)
ax1 = sns.distplot(df['price'], hist = False, color = 'r', label = 'Actual Value')
sns.distplot (Y_hat, hist = False, color = 'b', label = 'Calculated Values', ax = ax1)
plt.title ('Actual vs Calculated Values for Price')
plt.xlabel ('Price')
plt.ylabel ('Proportion of Cars')
plt.show()
```

Polynomial Fit

[ ]:
```
# function to plot data
def PlotPoly(model, x, y, Name):
  x_new = np.linspace(15, 55, 100)
  y_new = model(x_new)

  plt.plot(x, y, '.', x_new, y_new, '-')
  plt.title('Polynomial Fit with Matplotlib for Price ~ Length')
  ax = plt.gca()
  ax.set_facecolor((0.898, 0.898, 0.898))
  fig = plt.gcf()
  plt.xlabel(Name)
  plt.ylabel('Price of Cars')

  plt.show()
```

[ ]:
```
# fit a cubic polynomial
x = df['highway-mpg']
y = df['price']
f = np.polyfit (x, y, 3)
p = np.poly1d (f)
```

[ ]:
```
# plot the function
PlotPoly (p, x, y, 'highway-mpg')
```

[ ]:
```

```