Linear Regression

Concept Session

Demo 2.1 : Simple Linear Regression Analysis

Importing the required Python packages

In [ ]:

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

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

Reading the data

In [ ]:

```
weather_df=pd.read_csv('/content/drive/MyDrive/ML/DS2_C5_S2_WeatherHistory_Data_Concept.csv')
weather_df
```

Performing data preparation

Visualization of Data

In [ ]:

```
fig, ax = plt.subplots()
ax.set_xlabel('xlabel', fontsize=12)
ax.set_ylabel('ylabel', fontsize=12)
weather_df.plot.scatter(x = 'Humidity', y = 'Temperature (C)', s = 20, ax=ax);
plt.show()
```

Splitting data

In [ ]:

```
X=np.array(weather_df['Humidity']).reshape((-1, 1))
y=np.array(weather_df['Temperature (C)'])
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=200)
```

Training the model

In [ ]:

```
s_model = LinearRegression().fit(X_train, y_train)
s_r_sq=s_model.score(X_train, y_train)
```

In [ ]:

```
print('coefficient of determination:', s_r_sq)
print('intercept:', s_model.intercept_)
print('slope:', s_model.coef_)
```

Predict the model

In [ ]:

```
y_pred = s_model.predict(X_test)
print('predicted response:', y_pred, sep='\n')
```

Evaluating the model performance

In [ ]:

```
MSE=mean_squared_error(y_test, y_pred)
MAE=mean_absolute_error(y_test,y_pred)
RMSE = mean_squared_error(y_test, y_pred, squared=False)
print(MSE, MAE, RMSE)
```

In [ ]:

```
fig, ax = plt.subplots()
ax.set_xlabel('xlabel', fontsize=12)
ax.set_ylabel('ylabel', fontsize=12)
plt.ylim((0,30))
#weather_df_test=X_test

weather_df.plot.scatter(x = 'Humidity', y = 'Temperature (C)', s = 30, ax=ax)
plt.plot(X_test, y_pred, color="blue", linewidth=2)
plt.show()
```

Demo 2.2 : Multiple Linear Regression Analysis

Performing data preparation

In [ ]:

```
X_2 = weather_df[['Humidity', 'Wind Speed (km/h)']].values.reshape(-1,2)
Y =weather_df['Temperature (C)']
x = X_2[:, 0]
y = X_2[:, 1]
z = Y
```

Visualization of data

In [ ]:

```
# Creating figure
fig = plt.figure(figsize = (16, 9))
ax = plt.axes(projection ="3d")

# Add x, y gridlines
ax.grid(b = True, color ='grey',
        linestyle ='-.', linewidth = 0.3,
        alpha = 0.2)


# Creating plot
ax.scatter3D(x, y, z)


ax.set_xlabel('Humidity', fontweight ='bold')
ax.set_ylabel('Wind Speed', fontweight ='bold')
ax.set_zlabel('Temperature', fontweight ='bold')


# show plot
plt.show()
```

Splitting the data

In [ ]:

```
X3=weather_df[['Humidity','Pressure (millibars)','Wind Speed (km/h)']].values.reshape(-1,3)
z3=np.array(z)
X_train,X_test,z_train,z_test=train_test_split(X3,z3,test_size=0.3,random_state=200)
```

Training the model

In [ ]:

```
model_mul = LinearRegression().fit(X_train, z_train)
```

In [ ]:

```
print('Intercept: \n', model_mul.intercept_)
print('Coefficients: \n', model_mul.coef_)
r_sq2 = model_mul.score(X_train,z_train)
print('coefficient of determination:', r_sq2)
```

Evaluating the model performance

In [ ]:

```
# MSE=mean_squared_error(z_test, model_mul.predict(X_test))
RMSE = mean_squared_error(z_test, model_mul.predict(X_test), squared=False)
print(MSE, RMSE)
```

In [ ]:

```
from mpl_toolkits.mplot3d import Axes3D

x_pred = np.linspace(0, 1, 15)    # range of porosity values
z_pred = np.linspace(0, 32, 15)   # range of brittleness values
xx_pred, zz_pred = np.meshgrid(x_pred, z_pred)
model_viz = np.array([xx_pred.flatten(), zz_pred.flatten()]).T
ols = LinearRegression()
model = ols.fit(X_2, Y)
r2 = model.score(X_2, Y)
predicted = model.predict(model_viz)
fig = plt.figure(figsize=(12, 6))

ax1 = fig.add_subplot(131, projection='3d')
ax2 = fig.add_subplot(132, projection='3d')
ax3 = fig.add_subplot(133, projection='3d')

axes = [ax1, ax2, ax3]

for ax in axes:
    ax.plot(x, y, z, color='k', zorder=15, linestyle='none', marker='o', alpha=0.5)
    ax.scatter(xx_pred.flatten(), zz_pred.flatten(), model.predict(model_viz), facecolor=(0,0,0,0), s=2
0, edgecolor='#70b3f0')
    ax.set_xlabel('Humidity', fontsize=12)
    ax.set_ylabel('Wind Speed', fontsize=12)
    ax.set_zlabel('Temperature', fontsize=12)
    ax.locator_params(nbins=4, axis='x')
    ax.locator_params(nbins=5, axis='x')

ax1.view_init(elev=28, azim=120)
ax2.view_init(elev=4, azim=114)
ax3.view_init(elev=60, azim=165)

fig.suptitle('$R^2 = %.2f$' % r2, fontsize=20)

fig.tight_layout()
```

In [ ]: