linearreg

September 26, 2024

[15]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[19]:
```
csv_file = 'AirQuality.csv'
df = pd.read_csv(csv_file, sep=',')
```

[6]:
```
df = df.drop(columns=['Unnamed: 15', 'Unnamed: 16'], axis=1)
```

[23]:
```
df = df.dropna()
```

[24]:
```
dt_series = pd.Series(data = [item.split("/")[2] + "-" + item.split("/")[1] +␣
 ↪"-" + item.split("/")[0] for item in df['Date']], index=df.index) + ' ' + pd.
 ↪Series(data=[str(item).replace(".", ":") for item in df['Time']], index=df.
 ↪index)
dt_series = pd.to_datetime(dt_series)
```

[25]:
```
df = df.drop(columns=['Date', 'Time'], axis=1)
df.insert(loc=0, column='DateTime', value=dt_series)
```

[26]:
```
year_series = dt_series.dt.year
```

[27]:
```
month_series = dt_series.dt.month

day_series = dt_series.dt.day
```

[28]:
```
day_name_series = dt_series.dt.day_name()
```

[29]:
```
df['Year'] = year_series
df['Month'] = month_series
df['Day'] = day_series
df['Day Name'] = day_name_series
```

[30]:
```
df = df.sort_values(by='DateTime')
```

[32]:

```
def comma_to_period(series):
    new_series = pd.Series(data=[float(str(item).replace(',', '.')) for item in␣
 ↪series], index=df.index)
    return new_series
```

[34]:

```
cols_to_correct = ['CO(GT)', 'C6H6(GT)', 'T', 'RH', 'AH']
for col in cols_to_correct:
    df[col] = comma_to_period(df[col])
```

[35]:

```
df = df.drop(columns=['NMHC(GT)', 'CO(GT)', 'NOx(GT)', 'NO2(GT)'], axis=1)
```

[36]:

```
aq_2004_df = df[df['Year'] == 2004]
aq_2005_df = df[df['Year'] == 2005]
```

[38]:

```
for col in aq_2004_df.columns[1:-4]:
  median = aq_2004_df.loc[aq_2004_df[col] != -200, col].median()
  aq_2004_df[col] = aq_2004_df[col].replace(to_replace=-200, value=median)
```

[39]:

```
for col in aq_2005_df.columns[1:-4]:
  median = aq_2005_df.loc[aq_2005_df[col] != -200, col].median()
  aq_2005_df[col] = aq_2005_df[col].replace(to_replace=-200, value=median)
```

[40]:

```
group_2004_month = aq_2004_df.groupby(by='Month')
group_2005_month = aq_2005_df.groupby(by='Month')
```

[41]:

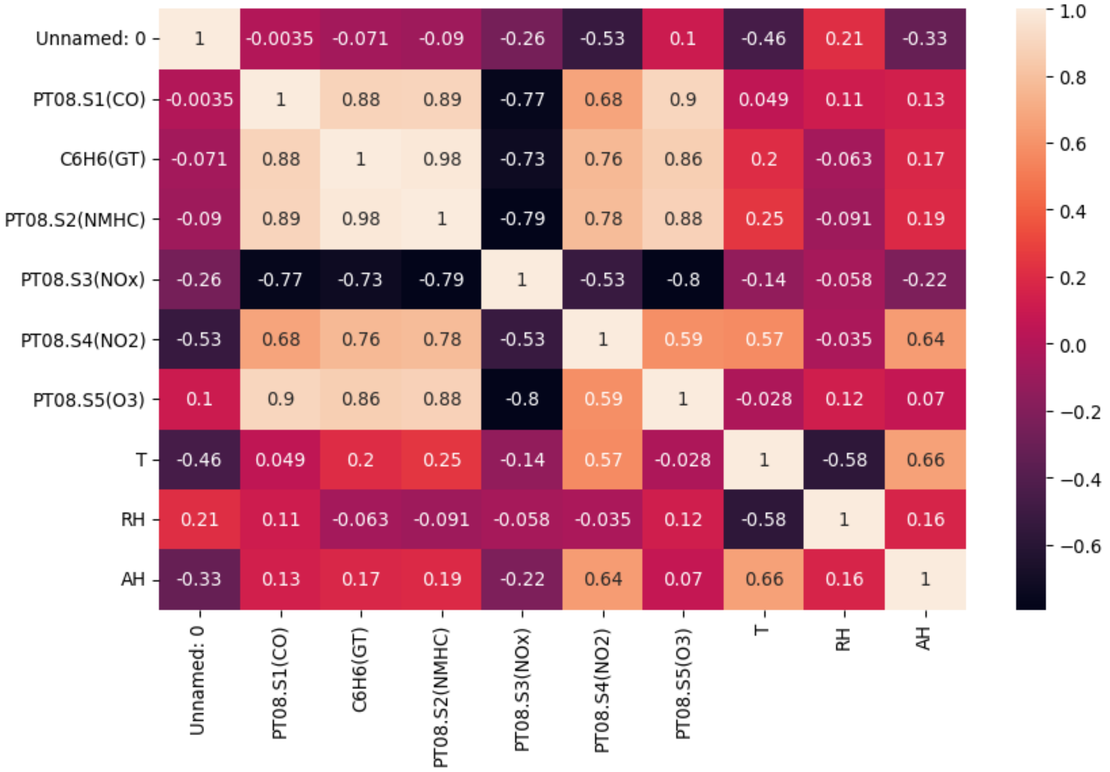
```
df = pd.concat([aq_2004_df, aq_2005_df])
```

[42]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 9357 entries, 0 to 9356
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   DateTime        9357 non-null   datetime64[ns]
 1   Unnamed: 0      9357 non-null   int64
 2   PT08.S1(CO)     9357 non-null   float64
 3   C6H6(GT)        9357 non-null   float64
 4   PT08.S2(NMHC)   9357 non-null   float64
 5   PT08.S3(NOx)    9357 non-null   float64
 6   PT08.S4(NO2)    9357 non-null   float64
 7   PT08.S5(O3)     9357 non-null   float64
 8   T               9357 non-null   float64
 9   RH              9357 non-null   float64
 10  AH              9357 non-null   float64
 11  Year            9357 non-null   int32
 12  Month           9357 non-null   int32
 13  Day             9357 non-null   int32
 14  Day Name        9357 non-null   object
dtypes: datetime64[ns](1), float64(9), int32(3), int64(1), object(1)
memory usage: 1.0+ MB
```

[43]:
```python
corr_df = df.iloc[:, 1:-4].corr()
plt.figure(figsize = (10, 6), dpi = 96)
sns.heatmap(data = corr_df, annot = True) # 'annot=True' fills the R values in the heatmap cells.
plt.show()
```



[44]:
```python
from sklearn.model_selection import train_test_split

X = df['T'] # Pandas DataFrame containing only feature variables
y = df['RH'] # Pandas Series containing the target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 42)
```

[45]:
```python
def errors_product():
  prod = (X_train - X_train.mean()) * (y_train - y_train.mean())
  return prod
```

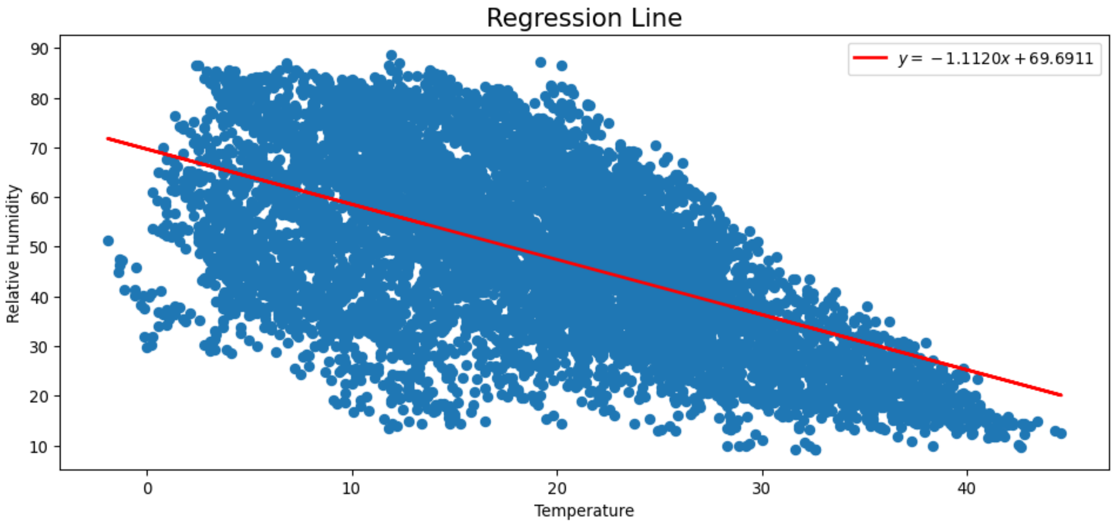
[46]:
```
def squared_errors():
  sq_errors = (X_train - X_train.mean()) ** 2
  return sq_errors
```

[47]:
```
slope = errors_product().sum()/ squared_errors().sum()
intercept = y_train.mean() - slope * X_train.mean()

print(f"Slope: {slope} \nIntercept: {intercept}")
```

```
Slope: -1.112053910794772
Intercept: 69.69110324644876
```

[48]:
```
plt.figure(figsize = (12, 5), dpi = 96)
plt.title("Regression Line", fontsize = 16)
plt.scatter(df['T'], df['RH'])
plt.plot(df['T'], slope * df['T'] + intercept, color = 'r', linewidth = 2,␣
 ↪label = '$y = -1.1120x + 69.6911$')
plt.xlabel("Temperature")
plt.ylabel("Relative Humidity")
plt.legend()
plt.show()
```



[49]:
```
from sklearn.linear_model import LinearRegression

X_train_reshaped = X_train.values.reshape(-1, 1)
y_train_reshaped = y_train.values.reshape(-1, 1)
X_test_reshaped = X_test.values.reshape(-1, 1)
y_test_reshaped = y_test.values.reshape(-1, 1)
```

```
lin_reg = LinearRegression()
lin_reg.fit(X_train_reshaped, y_train_reshaped)

print("Coefficient of $x$ (or slope) ==>", lin_reg.coef_)
print("Intercept ==>", lin_reg.intercept_)
```

```
Coefficient of $x$ (or slope) ==> [[-1.11205391]]
Intercept ==> [69.69110325]
```

[50]:

```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error

y_train_pred = lin_reg.predict(X_train_reshaped)
y_test_pred = lin_reg.predict(X_test_reshaped)

print(f"Train Set\n{'-' * 50}")
print(f"R-squared: {r2_score(y_train_reshaped, y_train_pred):.3f}")
print(f"Mean Squared Error: {mean_squared_error(y_train_reshaped, y_train_pred):
  ↪.3f}")
print(f"Root Mean Squared Error: {np.sqrt(mean_squared_error(y_train_reshaped,␣
  ↪y_train_pred)):.3f}")
print(f"Mean Absolute Error: {mean_absolute_error(y_train_reshaped,␣
  ↪y_train_pred):.3f}")

print(f"\n\nTest Set\n{'-' * 50}")
print(f"R-squared: {r2_score(y_test_reshaped, y_test_pred):.3f}")
print(f"Mean Squared Error: {mean_squared_error(y_test_reshaped, y_test_pred):.
  ↪3f}")
print(f"Root Mean Squared Error: {np.sqrt(mean_squared_error(y_test_reshaped,␣
  ↪y_test_pred)):.3f}")
print(f"Mean Absolute Error: {mean_absolute_error(y_test_reshaped, y_test_pred):
  ↪.3f}")
```

```
Train Set
--------------------------------------------------
R-squared: 0.325
Mean Squared Error: 195.281
Root Mean Squared Error: 13.974
Mean Absolute Error: 11.289


Test Set
--------------------------------------------------
R-squared: 0.346
Mean Squared Error: 187.026
Root Mean Squared Error: 13.676
Mean Absolute Error: 11.150
```