Fall 2023 DATA 220 Mathematical Methods for Data Analytics

Homework – 5

Name :- Prayag Nikul Purani

SJSU Id :- 017416737

Problem 1:
Importing the file

```
In [1]: import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression
        from sklearn.metrics import mean_squared_error, r2_score
```

```
In [2]: df = pd.read_csv('data.csv')
```

Splitting the dataset into test and train data.

Split of data set

```
In [5]: # Extract features (X) and target (y)
        X = df.iloc[:, :81]  # Features
        y = df['critical_temp']   # Target
```

```
In [6]: # Split the dataset into training and test sets
        random_seed = 2023
        test_size = 0.20
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=test_size, random_state=random_seed)
```

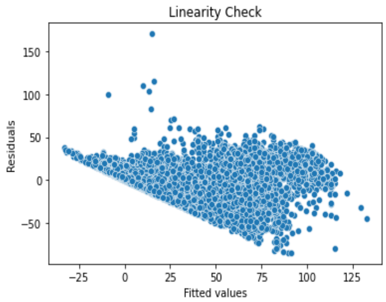
(A) Five Assumptions

A. Linearity:
There should be a linear connection between the dependent variable (target) and the independent variables (features). This implies that adjustments to the independent variables ought to cause a corresponding adjustment to the dependent variable.

Interpretation: To verify that the connection is generally linear, plot each independent variable against the dependent variable. This will allow you to verify the assumption.

```
In [9]: # Check assumptions
        # 1. Linearity
        # Scatter plot of residuals vs. predicted values
        sns.scatterplot(model.fittedvalues, model.resid)
        plt.xlabel('Fitted values')
        plt.ylabel('Residuals')
        plt.title('Linearity Check')
        plt.show()
```

```
C:\Users\Prayag Purani\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as ke
yword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

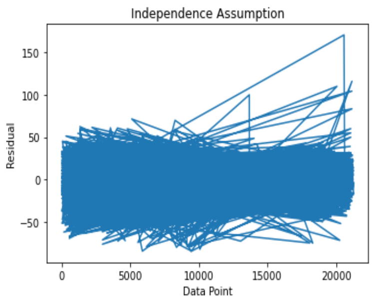


B. Independence:

The disparities between the observed and expected values, or residuals, need to be unrelated to one another. The residuals shouldn't show any consistent trends.

Interpretation: Look for residual autocorrelation. Finding patterns can be aided by plotting residuals versus time or any other pertinent variable.

```
In [10]: # 2. Independence of residuals
         residuals = y_train - X_train @ theta
         plt.plot(residuals)
         plt.xlabel('Data Point')
         plt.ylabel('Residual')
         plt.title('Independence Assumption')
         plt.show()
```

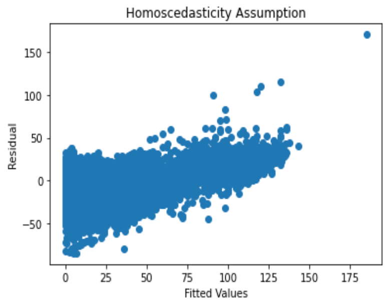


C. Homoscedasticity:

All levels of the independent variables should have a consistent variance in the residuals. Stated otherwise, there should be an approximately constant residual spread.

Plot the residuals against the expected values to interpret the results. The assumption is satisfied if the residual spread stays mostly constant. A discernible pattern, like the form of a funnel, might point to heteroscedasticity.

```
In [11]: # 3. Homoscedasticity (Constant Variance)
plt.scatter(y_train, residuals)
plt.xlabel('Fitted Values')
plt.ylabel('Residual')
plt.title('Homoscedasticity Assumption')
plt.show()
```

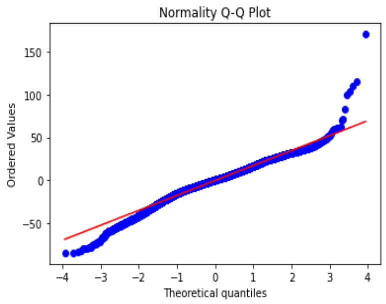


D. Normality:

A normal distribution should be seen in the residuals. For larger sample sizes, this assumption is not essential, but for lower sample sizes, normalcy is necessary to draw reliable statistical conclusions.

Interpretation: Plot the residuals as a Q-Q plot or as a histogram. The assumption is deemed met if the distribution is roughly normal.

```
In [12]: # 4. Normality of residuals
from scipy.stats import probplot
probplot(residuals, plot=plt)
plt.title('Normality Q-Q Plot')
plt.show()
```



E. No perfect Multicollinearity:

The independent variables shouldn't have perfect multicollinearity. That is to say, no independent variable ought to be a perfect linear combination of other variables.

Interpretation: For every independent variable, look up the variance inflation factor (VIF). More than ten VIF values are frequently seen as suggestive of multicollinearity.

In [13]:
```python
# 5. No perfect multicollinearity

from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = [variance_inflation_factor(X_train, i) for i in range(X_train.shape[1])]
print('VIF for each feature:')
print(vif)
# VIF should be less than 10 for all variables
# Interpretation: Based on the visualizations and tests, check if assumptions are met or not.
```

```
VIF for each feature:
[1719.7993635939167, 80.35255162607619, 426.0447605453024, 839.0227355438152, 455.27344781705773, 902.2125058907174, 196.006601
25549078, 148.3786250667996, 55.96343534998275, 24.26079001367438, 110.39939199098629, 83.97914657019359, 2105.3837707350285, 8
528.268913737405, 1646.8525563007174, 5808.350623030543, 4107.51629361195, 173.67852890369696, 268.5171586374379, 45.3418523919
70406, 416.5332904043671, 480.0789344582969, 904.774801830959, 3363.014345156241, 1091.7756281844327, 4983.626292478288, 2981.8
8990694724, 323.87150518171103, 151.2559235695471, 21.286686061177765, 349.1982923474105, 341.7618255067749, 138.128841842716,
269.9671349866302, 210.5949678066041, 372.99319692085663, 99.88615216360728, 50.97056619802191, 54.04071258577152, 27.329589396
285513, 94.85962623283218, 47.913976131400254, 113.9566242053223, 181.17662780225456, 94.68113365904394, 135.75563258973793, 5
4.986044703042054, 27.295942640484313, 71.70873044915865, 24.144400589677517, 110.85103792218803, 43.48097797380727, 315.119061
9579143, 527.5180321494138, 212.56026994594575, 386.8025081904225, 73.0369026838722, 35.66836668138936, 128.02403870090546, 41.
11956840697495, 362.34530872916974, 93.41007346797286, 62.59394403427714, 112.03328920972163, 44.86155184031164, 82.91711392064
877, 29.55860317236769, 18.051265953830608, 314.01712738964414, 34.38866228673739, 435.05304237889055, 153.820860542701, 2733.6
03545642491, 5001.0384882596645, 2440.9311507032576, 4266.473276992644, 1637.1391621922019, 306.21259288844396, 57.067007563818
97, 26.32309781840412, 96.64047531125462, 51.34310478410314]
```

(B) Derive the equation

→ <u>devire the equation</u>

formulás used →

$$\frac{\partial y}{\partial x} = 0 \;;\; \frac{\partial (Ax)}{\partial x} = A \;,\; \frac{\partial (xA)}{\partial x} = A^T \;;\; \frac{\partial (x^TAx)}{\partial x} = 2x^TA$$

$$\hat{y} = b_0 + b_1x_1 + b_2x_2 \text{------} \; b_nx_n$$

$$= b_0x_0 + b_1x_1 + \text{-------} \; b_nx_n$$

$$\hat{y} = \sum_{i=0}^{x=n} x_i b_i$$

error $= (y-\hat{y})^2$

$\Sigma e^2 = e^Te$ (from matrix)

$$= (y-\hat{y})^T(y-\hat{y})$$

$$= (y-xb)^T(y-xb)$$

$$= (y^T - b^Tx^T)(y-xb)$$

$$J = y^Ty - y^Txb - b^Tx^Ty + b^Tx^Txb$$

$$\frac{\partial J}{\partial b} = 0 - y^Tx - (x^Ty)^T + 2b^Tx^Tx$$

from here we need to find critical point

$$\frac{\partial J}{\partial b} = 0$$

$$\Rightarrow -y^Tx - y^Tx + 2b^Tx^Tx = 0$$

$$\Rightarrow -2y^Tx + 2b^Tx^Tx = 0$$

$$\Rightarrow b^Tx^Tx = y^Tx$$

$$b^T (x^T x)(x^T x)^{-1} = y^T x (x^T x)^{-1}$$

$$b^T = y^T x (x^T x)^{-1}$$

$$(b^T)^T = (y^T x (x^T x)^{-1})^T$$

$$\boxed{b_{opt} = (x^T x)^{-1} x^T y}$$

(C) Standardization technique

```
In [16]: from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler()

In [17]: X_train_scaled = scaler.fit_transform(X_train)

In [18]: X_test_scaled = scaler.transform(X_test)
```

(D) Optimal value

1.d Optimal values of intercept

In [19]:
```
import numpy as np

# Calculate the coefficients using the normal equation
XtX_inv = np.linalg.pinv(X_train_scaled.T @ X_train_scaled)
Xty = X_train_scaled.T @ y_train
beta_optimal = XtX_inv @ Xty
```

In [20]:
```
beta_optimal
```

Out[20]:
```
array([  -4.69389941,   28.39967588,  -34.54166212,  -18.50284048,
         27.43008693,  -11.72234227,    1.2177059 ,   12.51078731,
          0.76411668,  -12.57153325,    2.88307004,   11.6734808 ,
        -25.6337268 ,   -9.49991357,   23.30780151,  -45.79015162,
         14.26269082,   20.67842769,    5.19991763,  -21.00807323,
         -2.86161686,  -10.88433837,   93.27802545,    4.62217687,
       -101.87293661,   30.1348673 ,   18.09896967,   13.74068673,
         -3.37098898,   -8.76831049,   -7.79010224,  -13.94004328,
          0.45058833,    4.01841449,    9.48564274,    4.19656964,
         -6.0475069 ,   -7.05149413,   -0.43436155,   10.37497201,
         -2.50320164,   -3.14806736,   16.24923833,    5.37959114,
        -17.93810357,    1.30887266,   -6.00707712,  -20.9186652 ,
         -3.65238182,   26.33105746,  -10.96230426,   18.58645616,
        -27.48511627,  -15.37758784,   21.88969331,   -7.27139551,
          9.37694102,   -7.91053125,    6.37935361,   -3.78832821,
          5.47913881,   -3.48540924,   24.71535763,   -0.9843336 ,
        -14.30249732,    4.38028842,    0.3160262 ,  -15.54113183,
         -9.72230471,   20.15048062,   -1.61863079,  -19.95592061,
         30.30644105,   24.50021876,  -34.73845843,   28.45894657,
        -26.21478034,    5.92214768,   -0.68501902,    3.49187505,
        -11.26687205])
```

(E) Find y predict

1.e Find Y predict

In [21]:
```
#X_test_scaled = sm.add_constant(X_test_scaled)
y_pred = X_test_scaled @ beta_optimal

# Create a dataframe with y_actual and ŷ_predict
results_df = pd.DataFrame({'y_actual': y_test, 'ŷ_predict': y_pred})
```

In [22]:
```
results_df
```

Out[22]:

	y_actual	ŷ_predict
2862	77.80	27.813682
13507	2.00	-22.660371
4350	28.10	32.105373
14758	0.02	-32.390831
6736	61.10	33.378494
...	...	...
9271	68.00	36.634057
15516	2.77	-28.632659
11935	3.64	-27.710088
16352	6.20	-36.769094
13204	3.60	-28.403930

4253 rows × 2 columns

(F) R2 and MSE

1.f R2 and MSE

In [23]:
```
# Predict using the optimal coefficients
y_pred = X_test_scaled @ beta_optimal

# Calculate R-squared
ssr = np.sum((y_test - y_pred)**2)
sst = np.sum((y_test - np.mean(y_test))**2)
r_squared = 1 - (ssr / sst)

# Calculate MSE
mse = np.mean((y_test - y_pred)**2)

# Interpretation
print(f"R-squared: {r_squared}")
print(f"MSE: {mse}")
```

```
R-squared: -0.26705600327785284
MSE: 1493.4528548260976
```

References: -

https://gocardless.com/en-us/guides/posts/multiple-linear-regression-mlr-definition/#:~:text=Multiple%20linear%20regression%2C%20shortened%20to,variables%20and%20single%20dependent%20variables.

https://www.investopedia.com/terms/m/mlr.asp

https://online.stat.psu.edu/stat462/node/83/

https://www.fireblazeaischool.in/blogs/assumptions-of-linear-regression/

https://www.statology.org/multiple-linear-regression-assumptions/

https://usq.pressbooks.pub/statisticsforresearchstudents/chapter/multiple-regression-assumptions/

https://clas.ucdenver.edu/marcelo-perraillon/sites/default/files/attached-files/week_7_diagnostics_0.pdf

http://www.ce.memphis.edu/7012/L15_MultipleLinearRegression_I.pdf

https://stats.stackexchange.com/questions/250730/what-is-the-mathematical-relationship-between-r2-and-mse

https://vitalflux.com/mean-square-error-r-squared-which-one-to-use/

https://www.bmc.com/blogs/mean-squared-error-r2-and-variance-in-regression-analysis/

https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e