Notebook

February 11, 2025

[24]:

```
```

[53]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
import statsmodels.api as sm
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVC
from sklearn.feature_selection import VarianceThreshold
import shap
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
from statsmodels.discrete.discrete_model import MNLogit
from numpy.linalg import LinAlgError
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, confusion_matrix, roc_auc_score
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from statsmodels.tsa.holtwinters import Holt
from sklearn.metrics import mean_squared_error
```

[ ]:

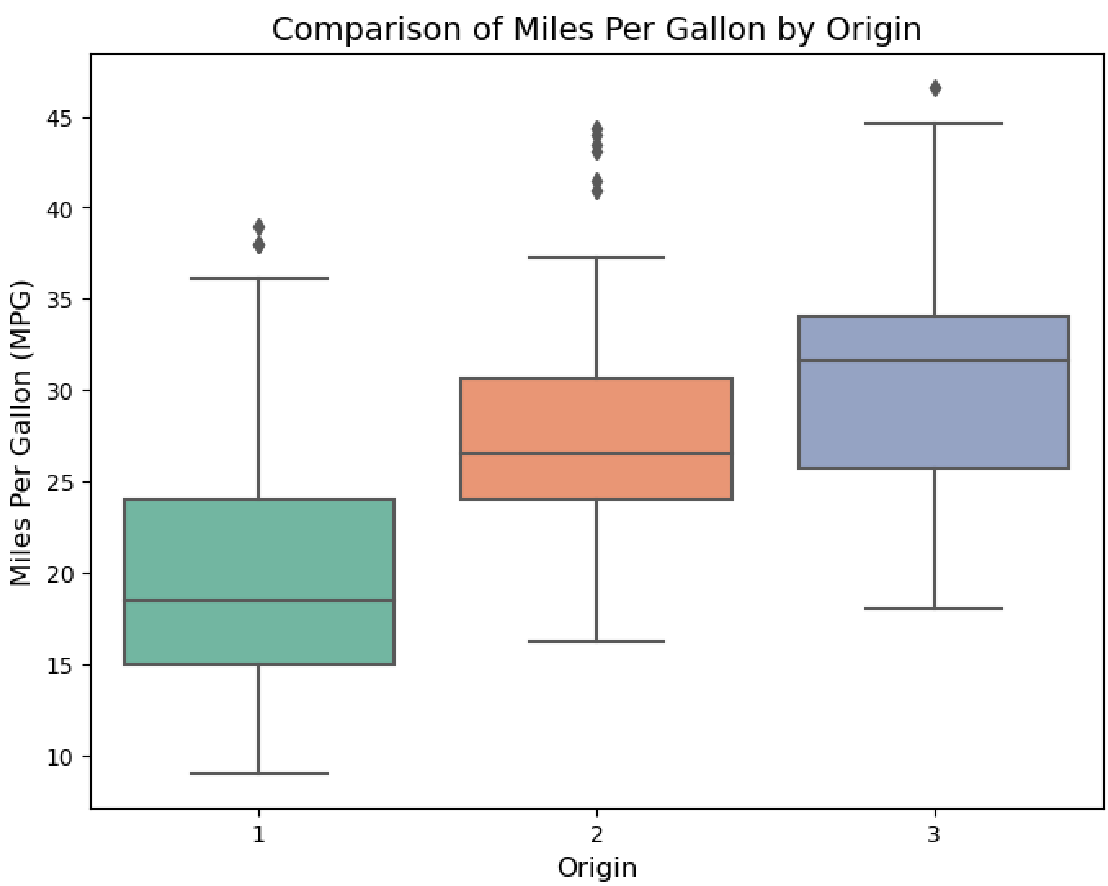
```
```

[54]:

```
import os

# Get absolute path of the current notebook directory
BASE_DIR = os.path.join(os.getcwd(), "data")

# Function to get the full path
```

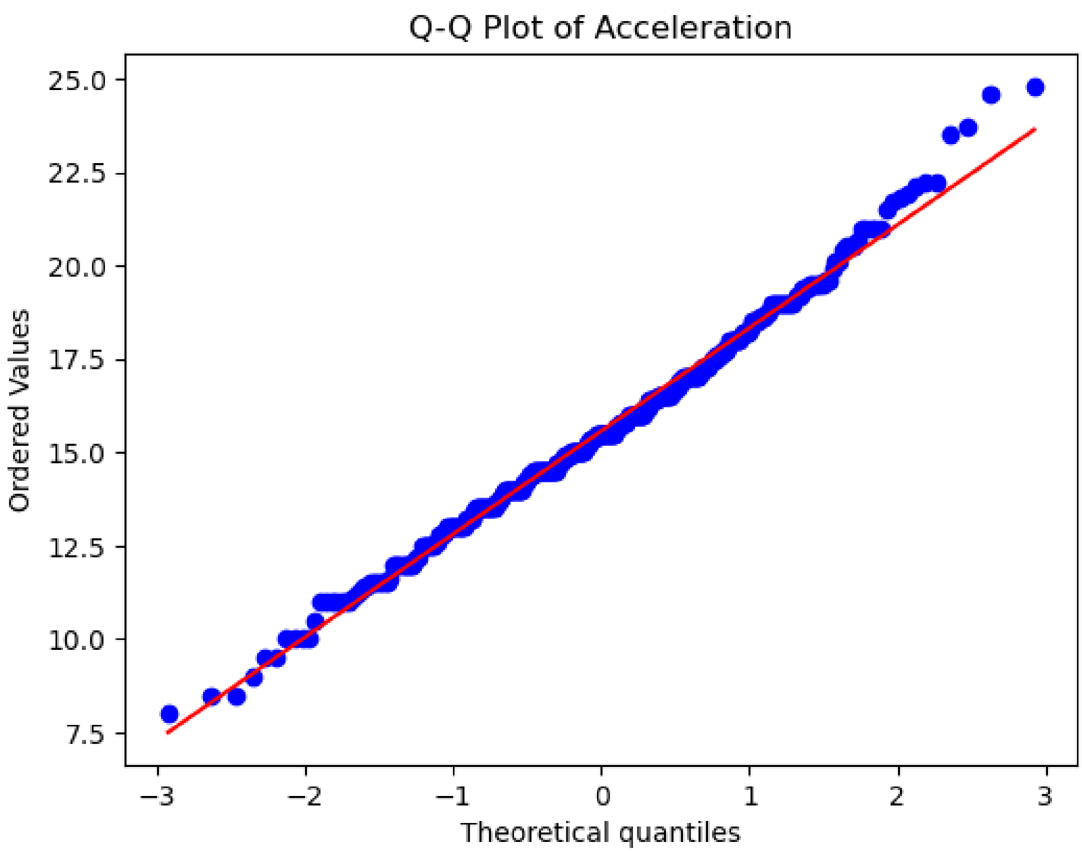
```python
def get_file_path(filename):
    return os.path.join(BASE_DIR, filename)
```

[55]:
```python
#load dataset
df= pd.read_csv(get_file_path("2025_Box-Plot_Dataset.csv"))

#creating a boxplot comparing Miles_Per_Gallon for different orign groups
plt.figure(figsize=(8,6))
sns.boxplot(x='Origin', y="Miles_Per_Gallon", data=df, palette="Set2")

plt.title("Comparison of Miles Per Gallon by Origin", fontsize=14)
plt.xlabel("Origin", fontsize=12)
plt.ylabel("Miles Per Gallon (MPG)", fontsize=12)

plt.show()
```



1 Q-Q Plot

[56]:
```
# Q-Q plot to check if 'Acceleration' follows the normal distribution

stats.probplot(df['Acceleration'], dist="norm", plot=plt)
plt.title("Q-Q Plot of Acceleration")
plt.show()
```



1.1 Kolmogorov-Smirnov (K-S) Test

[57]:
```
ks_stat, ks_p_value = stats.kstest(df['Acceleration'], 'norm',␣
↪args=(df['Acceleration'].mean(), df['Acceleration'].std()))

print(f"Kolmogorov-Smirnov Test: Statistic={ks_stat:.4f}, p-value={ks_p_value:.
↪4f}")

alpha = 0.05 #level of significance

if ks_p_value <= alpha:
```

```
    print("K-S Test: Reject H0 → 'Acceleration' is NOT normally distributed.")
else:
    print("K-S Test: Fail to reject H0 → 'Acceleration' follows a normal␣
  ↪distribution.")
```

```
Kolmogorov-Smirnov Test: Statistic=0.0508, p-value=0.2466
K-S Test: Fail to reject H0 → 'Acceleration' follows a normal distribution.
```

1.2 Shapiro-Wilk Test

[58]:
```
shapiro_stat, shapiro_p_value = stats.shapiro(df['Acceleration'])

print(f"Shapiro-Wilk Test: Statistic={shapiro_stat:.4f},␣
  ↪p-value={shapiro_p_value:.4f}")

alpha = 0.05  # Significance level

if shapiro_p_value < alpha:
    print("S-W Test: Reject H0 → 'Acceleration' is NOT normally distributed.")
else:
    print("S-W Test: Fail to reject H0 → 'Acceleration' follows a normal␣
  ↪distribution.")
```

```
Shapiro-Wilk Test: Statistic=0.9924, p-value=0.0399
S-W Test: Reject H0 → 'Acceleration' is NOT normally distributed.
```

1.3 Linear Regression Model

[ ]:
```

```

[59]:
```
df = pd.read_csv(get_file_path('2025_Regression_Dataset.csv'))

# Display first few rows
print(df.head())
```

```
   wtd_mean_atomic_radius  wtd_gmean_ThermalConductivity  \
0              105.514286                       0.621979
1              106.342857                       0.624878
2              104.371429                       0.629441
3              104.542857                       0.633910
4              104.885714                       0.642942

   wtd_gmean_FusionHeat  wtd_gmean_fie  wtd_gmean_ElectronAffinity  \
0              1.040986     938.016780                   99.414682
1              1.044545     937.025573                   97.774719
2              1.039211     940.294344                   98.411962
3              1.040986     940.391699                   96.998357
4              1.044545     940.586438                   94.231770
```

```
   wtd_entropy_ThermalConductivity  wtd_entropy_FusionHeat  \
0                         0.262848                0.994998   
1                         0.272820                1.044970   
2                         0.283412                0.964031   
3                         0.295609                0.994998   
4                         0.316852                1.044970   

   wtd_entropy_ElectronAffinity  wtd_entropy_Density  std_FusionHeat  …  \
0                      0.787382             0.814598        4.599064  …   
1                      0.787396             0.859811        4.599064  …   
2                      0.777657             0.769628        4.599064  …   
3                      0.775688             0.791445        4.599064  …   
4                      0.771173             0.830563        4.599064  …   

   mean_ThermalConductivity  mean_ElectronAffinity  mean_Density  \
0                107.756645                81.8375    4654.35725   
1                107.756645                81.8375    4654.35725   
2                112.006645                79.6075    4434.35725   
3                112.006645                79.6075    4434.35725   
4                112.006645                79.6075    4434.35725   

   gmean_ThermalConductivity  gmean_FusionHeat   gmean_fie  gmean_atomic_mass  \
0                   7.062488          3.479475  718.152900          66.361592   
1                   7.062488          3.479475  718.152900          66.361592   
2                   8.339818          3.479475  734.219624          59.310096   
3                   8.339818          3.479475  734.219624          59.310096   
4                   8.339818          3.479475  734.219624          59.310096   

   entropy_ThermalConductivity  entropy_ElectronAffinity  critical_temp  
0                     0.308148                  1.159687           29.0  
1                     0.308148                  1.159687           23.0  
2                     0.403693                  1.096672           36.0  
3                     0.403693                  1.096672           31.0  
4                     0.403693                  1.096672           33.0  

[5 rows x 28 columns]
```

[60]:

```python
print(df.isnull().sum())

print(df.info())

print(df.describe())
```

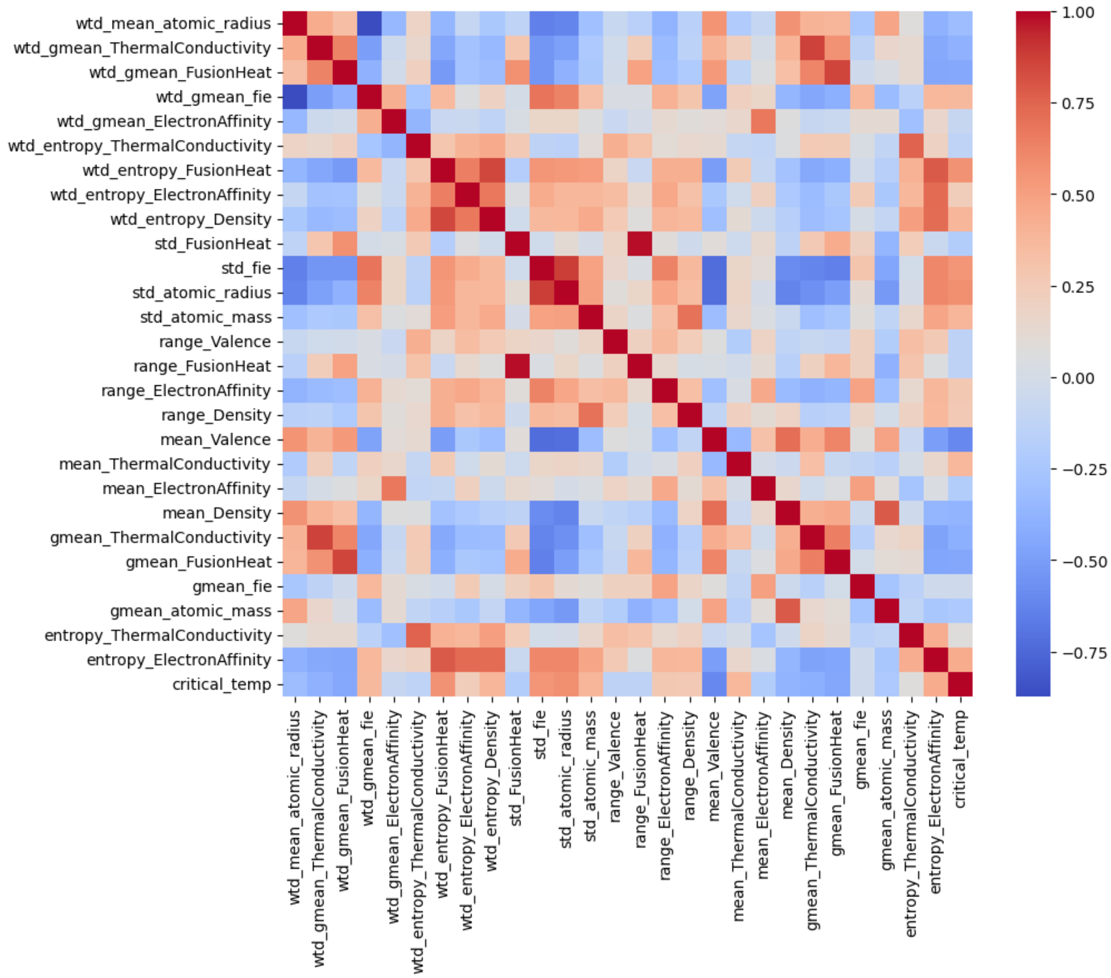
```
wtd_mean_atomic_radius             0
wtd_gmean_ThermalConductivity      0
wtd_gmean_FusionHeat               0
wtd_gmean_fie                      0
```

```
wtd_gmean_ElectronAffinity         0
wtd_entropy_ThermalConductivity    0
wtd_entropy_FusionHeat             0
wtd_entropy_ElectronAffinity       0
wtd_entropy_Density                0
std_FusionHeat                     0
std_fie                            0
std_atomic_radius                  0
std_atomic_mass                    0
range_Valence                      0
range_FusionHeat                   0
range_ElectronAffinity             0
range_Density                      0
mean_Valence                       0
mean_ThermalConductivity           0
mean_ElectronAffinity              0
mean_Density                       0
gmean_ThermalConductivity          0
gmean_FusionHeat                   0
gmean_fie                          0
gmean_atomic_mass                  0
entropy_ThermalConductivity        0
entropy_ElectronAffinity           0
critical_temp                      0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5275 entries, 0 to 5274
Data columns (total 28 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   wtd_mean_atomic_radius           5275 non-null   float64
 1   wtd_gmean_ThermalConductivity    5275 non-null   float64
 2   wtd_gmean_FusionHeat             5275 non-null   float64
 3   wtd_gmean_fie                    5275 non-null   float64
 4   wtd_gmean_ElectronAffinity       5275 non-null   float64
 5   wtd_entropy_ThermalConductivity  5275 non-null   float64
 6   wtd_entropy_FusionHeat           5275 non-null   float64
 7   wtd_entropy_ElectronAffinity     5275 non-null   float64
 8   wtd_entropy_Density              5275 non-null   float64
 9   std_FusionHeat                   5275 non-null   float64
 10  std_fie                          5275 non-null   float64
 11  std_atomic_radius                5275 non-null   float64
 12  std_atomic_mass                  5275 non-null   float64
 13  range_Valence                    5275 non-null   int64
 14  range_FusionHeat                 5275 non-null   float64
 15  range_ElectronAffinity           5275 non-null   float64
 16  range_Density                    5275 non-null   float64
 17  mean_Valence                     5275 non-null   float64
```

```
 18  mean_ThermalConductivity        5275 non-null   float64
 19  mean_ElectronAffinity           5275 non-null   float64
 20  mean_Density                    5275 non-null   float64
 21  gmean_ThermalConductivity       5275 non-null   float64
 22  gmean_FusionHeat                5275 non-null   float64
 23  gmean_fie                       5275 non-null   float64
 24  gmean_atomic_mass               5275 non-null   float64
 25  entropy_ThermalConductivity     5275 non-null   float64
 26  entropy_ElectronAffinity        5275 non-null   float64
 27  critical_temp                   5275 non-null   float64
dtypes: float64(27), int64(1)
memory usage: 1.1 MB
None
       wtd_mean_atomic_radius  wtd_gmean_ThermalConductivity  \
count             5275.000000                    5275.000000
mean               134.681841                      26.831423
std                 28.787768                      40.105763
min                 64.600000                       0.072768
25%                112.140667                       1.084912
50%                125.833333                       6.085087
75%                158.391200                      45.875927
max                253.000000                     358.713959

       wtd_gmean_FusionHeat  wtd_gmean_fie  wtd_gmean_ElectronAffinity  \
count           5275.000000    5275.000000                 5275.000000
mean               9.990091     833.516080                   72.219307
std               12.973263     119.359462                   31.490091
min                0.480799     502.500000                    1.500000
25%                1.321875     721.053648                   50.438290
50%                4.821853     858.874330                   72.854039
75%               16.393145     937.697803                   89.962916
max              105.000000    1183.712294                  214.651659

       wtd_entropy_ThermalConductivity  wtd_entropy_FusionHeat  \
count                      5275.000000             5275.000000
mean                          0.542039                0.922645
std                           0.319675                0.367980
min                           0.000000                0.000000
25%                           0.250750                0.679053
50%                           0.551852                1.000424
75%                           0.777388                1.163470
max                           1.584219                1.674166

       wtd_entropy_ElectronAffinity  wtd_entropy_Density  std_FusionHeat  …  \
count                   5275.000000          5275.000000     5275.000000  …
mean                       0.777248             0.862335        8.271298  …
std                        0.285434             0.318776        8.666517  …
min                        0.000000             0.000000        0.000000  …
```

```
25%                        0.666039             0.690637          4.261726  …
50%                        0.785771             0.892008          4.948155  …
75%                        0.879506             1.089535          8.935301  …
max                        1.675375             1.659095         51.635000  …

       mean_ThermalConductivity  mean_ElectronAffinity  mean_Density  \
count               5275.000000            5275.000000   5275.000000
mean                  89.798759              76.945910   6117.305246
std                   38.398988              27.810696   2866.393179
min                    0.115415               1.500000    535.000000
25%                   61.000000              61.713333   4529.571500
50%                   97.000000              73.100000   5329.085800
75%                  111.004430              85.512500   6642.000000
max                  332.500000             228.300000  22590.000000

       gmean_ThermalConductivity  gmean_FusionHeat    gmean_fie  \
count                5275.000000       5275.000000  5275.000000
mean                   29.281409         10.020024   738.018816
std                    33.079523          9.744749    77.991794
min                     0.072768          0.640000   490.633468
25%                     8.339818          4.141514   692.541331
50%                    14.287643          5.196603   728.828771
75%                    41.279535         13.481005   765.779664
max                   317.883627        105.000000  1225.900159

       gmean_atomic_mass  entropy_ThermalConductivity  \
count        5275.000000                  5275.000000
mean           71.218091                     0.729671
std            31.279883                     0.325210
min             5.320573                     0.000000
25%            58.041225                     0.457810
50%            66.361592                     0.741854
75%            77.118550                     0.962398
max           208.980400                     1.633977

       entropy_ElectronAffinity  critical_temp
count               5275.000000    5275.000000
mean                   1.077926      35.006786
std                    0.337212      34.121958
min                    0.000000       0.000500
25%                    0.894571       5.700000
50%                    1.158144      21.500000
75%                    1.349501      64.350000
max                    1.767732     136.000000

[8 rows x 28 columns]
```

```
[61]: # Featuyre selection => Perfom correlation analysis to check:
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=False, cmap='coolwarm')
plt.show()
```




```
[62]: #Selected indepedet variables based on corrrelation
selected_features␣
  ↪=["wtd_gmean_fie","wtd_entropy_FusionHeat","wtd_entropy_ElectronAffinity","wtd_entropy_Dens
  ↪"std_fie","std_atomic_radius", "std_atomic_mass",
"range_ElectronAffinity","range_Density","mean_ThermalConductivity","entropy_ElectronAffinity'


# Define independent (X) and dependent (y) variables
X = df[selected_features]
y = df["critical_temp"]
```

```
#split data into training & testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Initialize and train model
model = LinearRegression()
model.fit(X_train, y_train)
```

[62]: LinearRegression()

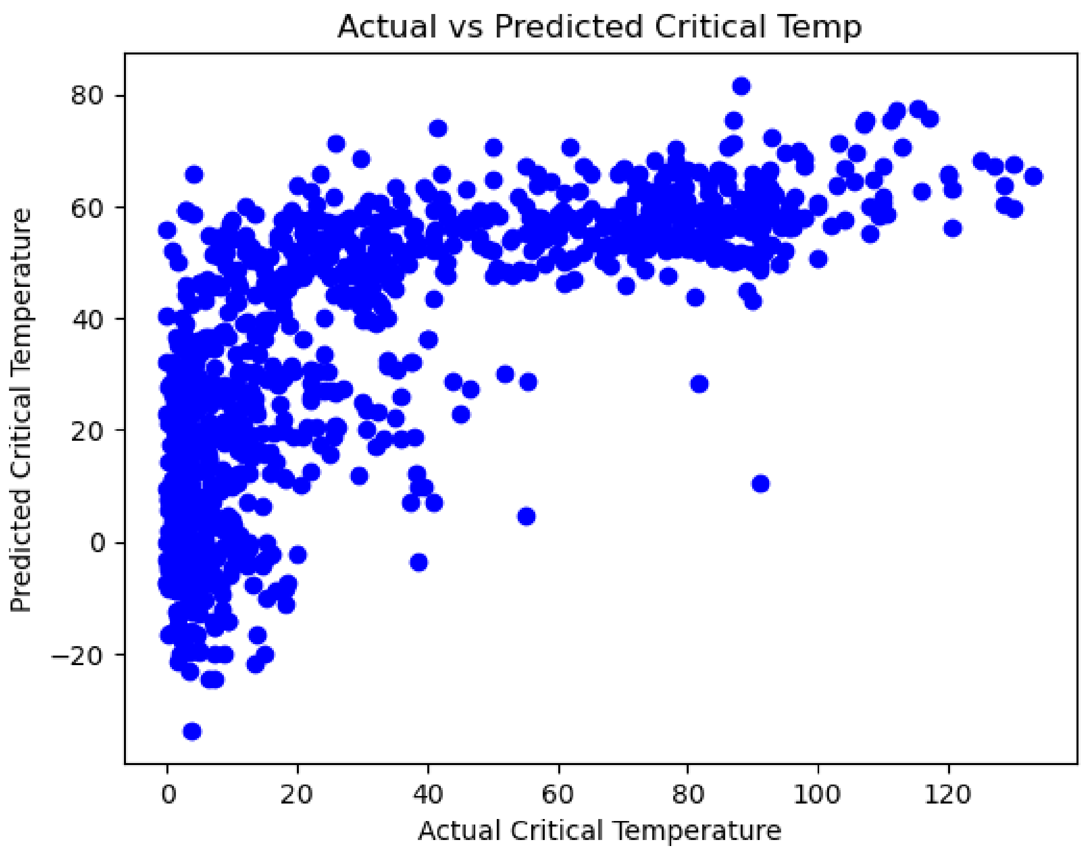
[63]:
```
print(f"Intercept: {model.intercept_}")
coefficients = pd.DataFrame({"Feature": selected_features, "Coefficient": model.
↪coef_})
print(coefficients)
```

```
Intercept: 9.858758417343662
                           Feature  Coefficient
0                     wtd_gmean_fie    -0.050728
1            wtd_entropy_FusionHeat    45.883388
2   wtd_entropy_ElectronAffinity   -29.566577
3               wtd_entropy_Density    -0.058841
4                           std_fie     0.091861
5                 std_atomic_radius     0.413353
6                   std_atomic_mass     0.126460
7            range_ElectronAffinity    -0.032388
8                     range_Density    -0.000680
9          mean_ThermalConductivity     0.179015
10       entropy_ElectronAffinity    -4.652724
```

[64]:
```
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared Score: {r2}")
```

```
Mean Squared Error: 569.6824682665567
R-squared Score: 0.5159699869885219
```

[65]:
```
plt.scatter(y_test, y_pred, color='blue')
plt.xlabel("Actual Critical Temperature")
plt.ylabel("Predicted Critical Temperature")
plt.title("Actual vs Predicted Critical Temp")
plt.show()
```



1.4 Logistic Regression Backward

[66]:
```
df = pd.read_csv(get_file_path("2025_Classification_Dataset.csv"))

# Data preprocessing
X = df.drop(columns=['class'])
y = df['class']

# Split the dataset first to prevent data leakage
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Handle categorical variables using LabelEncoder on training data and␣
 ↪transform test data
for col in X_train.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    # Fit on training data and transform both train and test
    X_train[col] = le.fit_transform(X_train[col].astype(str))
    # Handle unseen categories in test by assigning a default value (e.g., -1)
```

```
    # Note: This may require more sophisticated handling depending on the␣
↪dataset
    X_test[col] = le.transform(X_test[col].astype(str))

# Remove constant features from training data
selector = VarianceThreshold(threshold=0)
X_train_processed = selector.fit_transform(X_train)
selected_columns = X_train.columns[selector.get_support()]
X_train = pd.DataFrame(X_train_processed, columns=selected_columns,␣
↪index=X_train.index)
X_test = X_test[selected_columns]

# Add constant for statsmodels
X_train_sm = sm.add_constant(X_train)

# Backward Selection with intercept protection
def backward_elimination(X, y):
    X_opt = X.copy()
    while True:
        try:
            model = MNLogit(y, X_opt).fit(disp=0)
        except LinAlgError:
            # Handle singular matrix by removing the last added feature and␣
↪break
            break
        p_values = model.pvalues
        # Exclude 'const' from consideration
        p_values_filtered = p_values.drop('const', errors='ignore')
        if p_values_filtered.empty:
            break
        max_p_value = p_values_filtered.max().max()
        if max_p_value > 0.05:
            # Find the feature with the highest p-value across any class
            max_feature = p_values_filtered.max(axis=1).idxmax()
            X_opt = X_opt.drop(columns=[max_feature])
        else:
            break
    return X_opt.columns

selected_features = backward_elimination(X_train_sm, y_train)
# Ensure 'const' is included if present
if 'const' in X_train_sm.columns:
    selected_features = list(selected_features) + ['const']
X_train_selected = X_train_sm[selected_features]
X_test_selected = sm.add_constant(X_test, has_constant='add')[selected_features]

# Standardize features
```

```
scaler = StandardScaler()
X_train_selected_scaled = scaler.fit_transform(X_train_selected)
X_test_selected_scaled = scaler.transform(X_test_selected)

# Model Fitting with increased max_iter and solver
model = LogisticRegression(max_iter=1000, solver='lbfgs',␣
↪multi_class='multinomial')
model.fit(X_train_selected_scaled, y_train)

# Model Evaluation
y_pred = model.predict(X_test_selected_scaled)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/statsmodels/base/model.py:604: ConvergenceWarning: Maximum Likelihood
optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

```
Accuracy: 0.9415584415584416
Confusion Matrix:
 [[724   3]
 [ 42   1]]
Classification Report:
               precision    recall  f1-score   support

          No       0.95      1.00      0.97       727
         Yes       0.25      0.02      0.04        43

    accuracy                           0.94       770
   macro avg       0.60      0.51      0.51       770
weighted avg       0.91      0.94      0.92       770
```

[67]:
```
import numpy as np

# Get coefficients and intercept
intercept = model.intercept_
coefficients = model.coef_[0]  # model.coef_ returns a 2D array, we take the␣
↪first row

# Print results
print("Intercept:", intercept)
for feature, coef in zip(selected_features, coefficients):
    print(f"Feature: {feature}, Coefficient: {coef}, Exp(Coeff): {np.
↪exp(coef)}")
```

```
Intercept: [-1.62724828]
```

```
Feature: const, Coefficient: 0.0, Exp(Coeff): 1.0
Feature: shift, Coefficient: 0.24586199666849642, Exp(Coeff): 1.2787230933176406
Feature: gpuls, Coefficient: 0.167953831801793, Exp(Coeff): 1.182881997856601
Feature: nbumps2, Coefficient: 0.16101110715643788, Exp(Coeff):
1.1746980162960454
Feature: nbumps3, Coefficient: 0.17179863722619895, Exp(Coeff):
1.1874387031877787
Feature: const, Coefficient: 0.0, Exp(Coeff): 1.0
```

[ ]:

```

```

[68]:

```
df = pd.read_csv(get_file_path('2025_Credit-card-clients.csv'))

# Encode categorical variables
categorical_cols = df.select_dtypes(include=["object"]).columns
label_encoders = {}
for col in categorical_cols:
    label_encoders[col] = LabelEncoder()
    df[col] = label_encoders[col].fit_transform(df[col])

# Split features and target
X = df.drop(columns=["Y"])  # Replace 'target' with actual target column name
y = df["Y"]

# Scale numerical features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,␣
↪random_state=42)
```

1.5 Model 1: Support Vector Machine (SVM)

[ ]:

```
svm_model = SVC(kernel="rbf", probability=True)
svm_model.fit(X_train, y_train)

# Predictions
y_pred_svm = svm_model.predict(X_test)
y_prob_svm = svm_model.predict_proba(X_test)[:, 1]

# Evaluation
print("  SVM Results  ")
print("Accuracy:", accuracy_score(y_test, y_pred_svm))
print("Precision:", precision_score(y_test, y_pred_svm))
print("Recall:", recall_score(y_test, y_pred_svm))
print("F1 Score:", f1_score(y_test, y_pred_svm))
```

```python
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_svm))
print("ROC-AUC Score:", roc_auc_score(y_test, y_prob_svm))
print("=" * 50)
```

1.6 Model 2: CHAID Decision Tree

[22]:
```python
decision_tree = DecisionTreeClassifier(criterion="entropy", max_depth=5,␣
↪min_samples_split=10)  # CHAID Approximation
decision_tree.fit(X_train, y_train)

# Predictions
y_pred_tree = decision_tree.predict(X_test)
y_prob_tree = decision_tree.predict_proba(X_test)[:, 1]

# Evaluation
print("  Decision Tree (CHAID) Results  ")
print("Accuracy:", accuracy_score(y_test, y_pred_tree))
print("Precision:", precision_score(y_test, y_pred_tree))
print("Recall:", recall_score(y_test, y_pred_tree))
print("F1 Score:", f1_score(y_test, y_pred_tree))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_tree))
print("ROC-AUC Score:", roc_auc_score(y_test, y_prob_tree))
print("=" * 50)
```

```
  Decision Tree (CHAID) Results
Accuracy: 0.806
Precision: 0.6790123456790124
Recall: 0.3473684210526316
F1 Score: 0.4596100278551532
Confusion Matrix:
 [[1447   78]
 [ 310  165]]
ROC-AUC Score: 0.7404210526315789
==================================================
```

[23]:
```python
## Model 3: K-Nearest Neighbors (KNN)

knn_model = KNeighborsClassifier(n_neighbors=5, metric="euclidean")
knn_model.fit(X_train, y_train)

# Predictions
y_pred_knn = knn_model.predict(X_test)
y_prob_knn = knn_model.predict_proba(X_test)[:, 1]

# Evaluation
print("  KNN Results  ")
print("Accuracy:", accuracy_score(y_test, y_pred_knn))
```

```python
print("Precision:", precision_score(y_test, y_pred_knn))
print("Recall:", recall_score(y_test, y_pred_knn))
print("F1 Score:", f1_score(y_test, y_pred_knn))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_knn))
print("ROC-AUC Score:", roc_auc_score(y_test, y_prob_knn))
print("=" * 50)
```

```
Exception ignored on calling ctypes callback function: <function _ThreadpoolInfo
._find_modules_with_dl_iterate_phdr.<locals>.match_module_callback at
0x7feb9d97f9a0>
Traceback (most recent call last):
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 400, in match_module_callback
    self._make_module_from_path(filepath)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 515, in _make_module_from_path
    module = module_class(filepath, prefix, user_api, internal_api)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 606, in __init__
    self.version = self.get_version()
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 646, in get_version
    config = get_config().split()
AttributeError: 'NoneType' object has no attribute 'split'
Exception ignored on calling ctypes callback function: <function _ThreadpoolInfo
._find_modules_with_dl_iterate_phdr.<locals>.match_module_callback at
0x7feb9d97f9a0>
Traceback (most recent call last):
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 400, in match_module_callback
    self._make_module_from_path(filepath)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 515, in _make_module_from_path
    module = module_class(filepath, prefix, user_api, internal_api)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 606, in __init__
    self.version = self.get_version()
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 646, in get_version
    config = get_config().split()
AttributeError: 'NoneType' object has no attribute 'split'
```

```
 KNN Results
Accuracy: 0.769
Precision: 0.5239852398523985
Recall: 0.29894736842105263
F1 Score: 0.3806970509383378
Confusion Matrix:
```

```
[[1396  129]
 [ 333  142]]
ROC-AUC Score: 0.6747113028472821
==================================================
```

[28]:
```python
import shap
from sklearn.ensemble import RandomForestClassifier

# 1. Train a Random Forest CLASSIFIER (not Regressor)
rf_model = RandomForestClassifier(
    n_estimators=100,
    class_weight="balanced",  # Handle class imbalance
    random_state=42
)
rf_model.fit(X_train, y_train)

# 2. Use TreeExplainer (optimized for tree models)
explainer = shap.TreeExplainer(rf_model)
shap_values = explainer.shap_values(X_test)

# 3. Rank features by importance (using SHAP values for class 1)
shap_importance = pd.DataFrame({
    "Feature": X.columns,
    "SHAP Importance": np.abs(shap_values[1]).mean(axis=0)
}).sort_values("SHAP Importance", ascending=False)

print("Top Features by SHAP Importance:\n", shap_importance.head(10))

# 4. Plot SHAP summary (for class 1)
shap.summary_plot(shap_values[1], X_test, plot_type="bar")
```

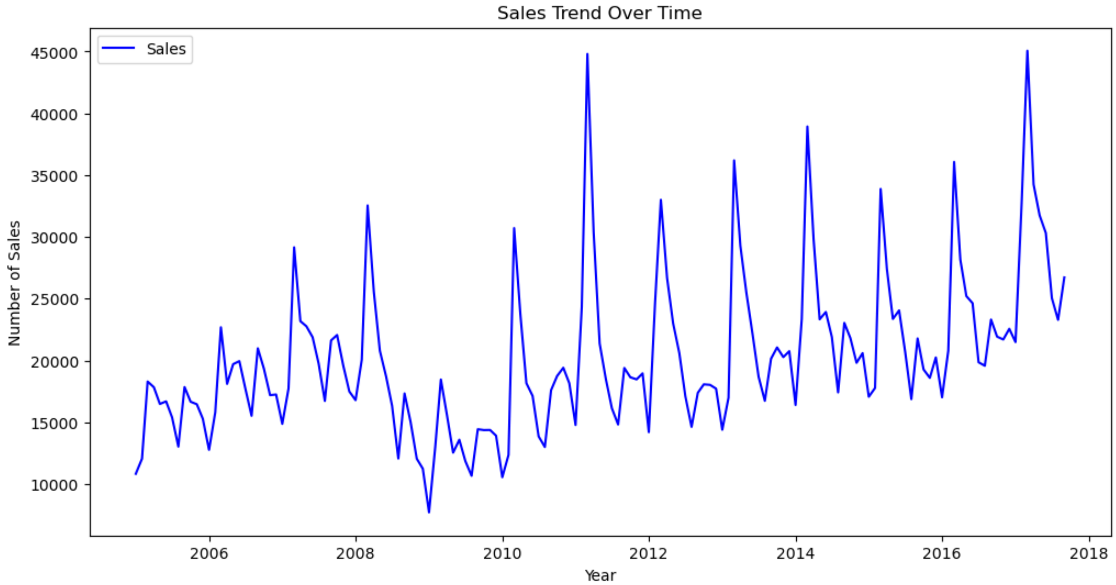
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[28], line 17
     14 shap_values = explainer.shap_values(X_test)
     16 # 3. Rank features by importance (using SHAP values for class 1)
---> 17 shap_importance = pd.DataFrame({
     18     "Feature": X.columns,
     19     "SHAP Importance": np.abs(shap_values[1]).mean(axis=0)
     20 }).sort_values("SHAP Importance", ascending=False)
     22 print("Top Features by SHAP Importance:\n", shap_importance.head(10))
     24 # 4. Plot SHAP summary (for class 1)

File ~/anaconda3/lib/python3.10/site-packages/pandas/core/frame.py:778, in
 ↪DataFrame.__init__(self, data, index, columns, dtype, copy)
    772     mgr = self._init_mgr(
    773         data, axes={"index": index, "columns": columns}, dtype=dtype,
 ↪copy=copy
```

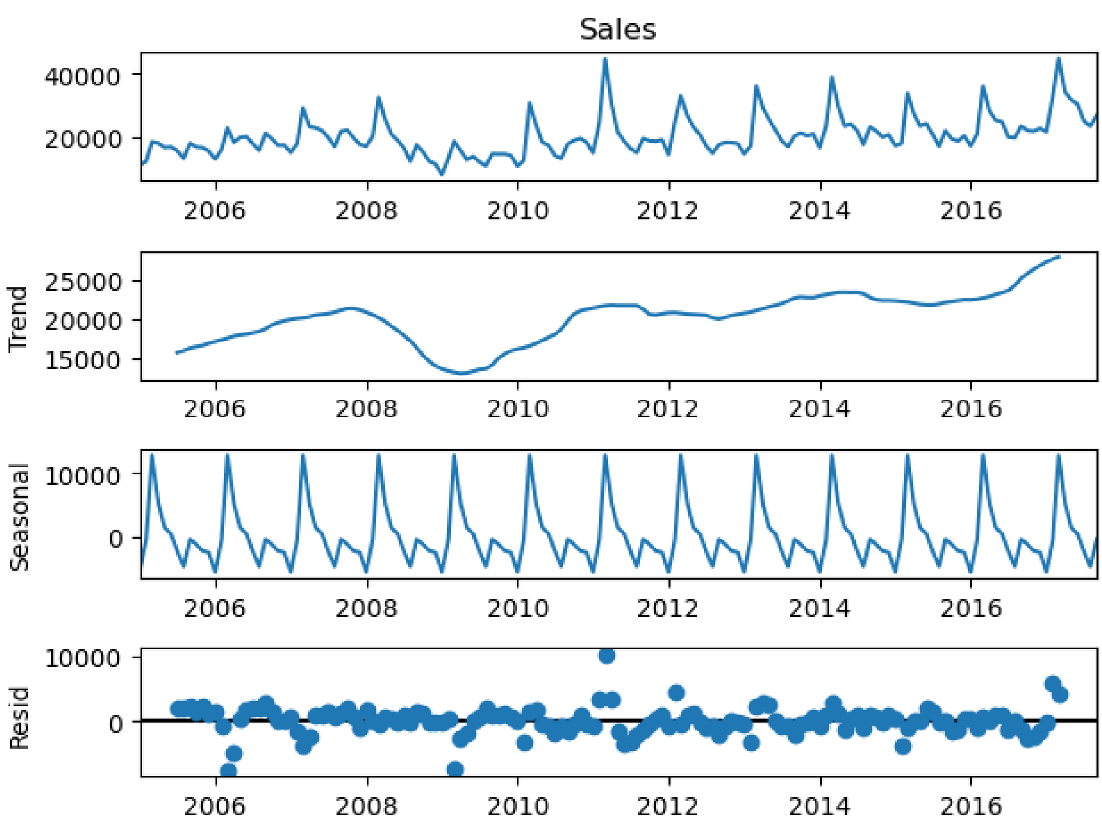
```
    774     )
    776 elif isinstance(data, dict):
    777     # GH#38939 de facto copy defaults to False only in non-dict cases
--> 778     mgr =␣
  ↪dict_to_mgr(data, index, columns, dtype=dtype, copy=copy, typ=manager)
    779 elif isinstance(data, ma.MaskedArray):
    780     from numpy.ma import mrecords

File ~/anaconda3/lib/python3.10/site-packages/pandas/core/internals/constructio .
  ↪py:503, in dict_to_mgr(data, index, columns, dtype, typ, copy)
    499     else:
    500         # dtype check to exclude e.g. range objects, scalars
    501         arrays = [x.copy() if hasattr(x, "dtype") else x for x in array ]
--> 503 return␣
  ↪arrays_to_mgr(arrays, columns, index, dtype=dtype, typ=typ, consolidate=copy)

File ~/anaconda3/lib/python3.10/site-packages/pandas/core/internals/constructio .
  ↪py:114, in arrays_to_mgr(arrays, columns, index, dtype, verify_integrity, typ ␣
  ↪consolidate)
    111 if verify_integrity:
    112     # figure out the index, if necessary
    113     if index is None:
--> 114         index = _extract_index(arrays)
    115     else:
    116         index = ensure_index(index)

File ~/anaconda3/lib/python3.10/site-packages/pandas/core/internals/constructio .
  ↪py:677, in _extract_index(data)
    675 lengths = list(set(raw_lengths))
    676 if len(lengths) > 1:
--> 677     raise ValueError("All arrays must be of the same length")
    679 if have_dicts:
    680     raise ValueError(
    681         "Mixing dicts with non-Series may lead to ambiguous ordering."
    682     )

ValueError: All arrays must be of the same length
```

[34]:
```
import pandas as pd

# Load dataset with correct delimiter and handle inconsistent formatting
df = pd.read_csv(get_file_path("TimeSeries_Dataset.csv"),
                 header=None,
                 names=["Year", "Month", "Sales"],
                 skipinitialspace=True)  # Removes extra spaces

# Forward fill missing years
```

```
df["Year"].fillna(method="ffill", inplace=True)

# Clean and convert 'Sales' column
df["Sales"] = df["Sales"].astype(str).str.replace(",", "").str.strip('" ').
↪astype(int)

# Create a 'Date' column
df["Date"] = pd.to_datetime(df["Year"].astype(int).astype(str) + "-" +␣
↪df["Month"].astype(int).astype(str), format="%Y-%m")

# Set 'Date' as index and drop unnecessary columns
df.set_index("Date", inplace=True)
df.drop(columns=["Year", "Month"], inplace=True)

# Plot sales over time
plt.figure(figsize=(12, 6))
plt.plot(df.index, df["Sales"], label="Sales", color="blue")
plt.title("Sales Trend Over Time")
plt.xlabel("Year")
plt.ylabel("Number of Sales")
plt.legend()
plt.show()

# Decomposition to analyze trend and seasonality
decomposition = sm.tsa.seasonal_decompose(df["Sales"], model='additive',␣
↪period=12)
decomposition.plot()
plt.show()
```

```
/tmp/ipykernel_12841/500565962.py:10: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df["Year"].fillna(method="ffill", inplace=True)
/tmp/ipykernel_12841/500565962.py:10: FutureWarning: Series.fillna with 'method'
is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill()
instead.
  df["Year"].fillna(method="ffill", inplace=True)
```

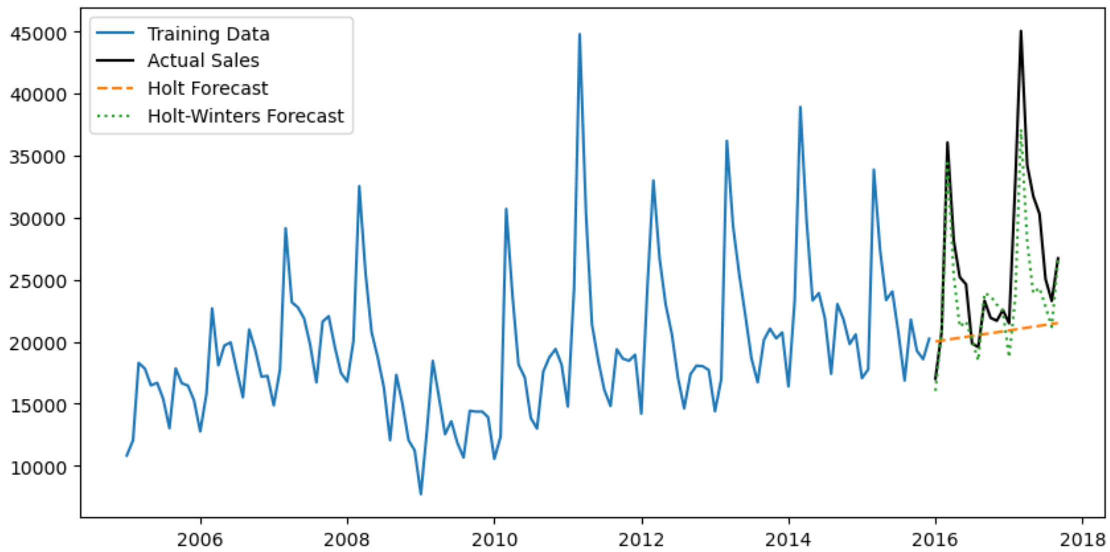

Sales Trend Over Time
Sales
45000
40000
35000
30000
25000
20000
15000
10000
Number of Sales
2006
2008
2010
2012
2014
2016
2018
Year


Sales
40000
20000
Trend
25000
20000
15000
Seasonal
10000
0
Resid
10000
0
2006
2008
2010
2012
2014
2016

[38]:

```python
# Splitting data
train = df[:'2015-12-01']
test = df['2016-01-01':]

# Holt's Linear Trend Model (HES)
holt_model = Holt(train['Sales']).fit()
holt_forecast = holt_model.forecast(len(test))
holt_mse = mean_squared_error(test['Sales'], holt_forecast)

# Holt-Winters Model (Multiplicative)
hw_model = ExponentialSmoothing(train['Sales'], trend='add', seasonal='add',␣
↪seasonal_periods=12).fit()
hw_forecast = hw_model.forecast(len(test))
hw_mse = mean_squared_error(test['Sales'], hw_forecast)

# Plotting
plt.figure(figsize=(10,5))
plt.plot(train.index, train['Sales'], label='Training Data')
plt.plot(test.index, test['Sales'], label='Actual Sales', color='black')
plt.plot(test.index, holt_forecast, label='Holt Forecast', linestyle='dashed')
plt.plot(test.index, hw_forecast, label='Holt-Winters Forecast',␣
↪linestyle='dotted')
plt.legend()
plt.show()

# Selecting the best model
best_model = "Holt-Winters" if hw_mse < holt_mse else "Holt"
print(f"MSE - Holt's Model: {holt_mse:.2f}")
print(f"MSE - Holt-Winters Model: {hw_mse:.2f}")
print(f"Recommended Model: {best_model}")
```

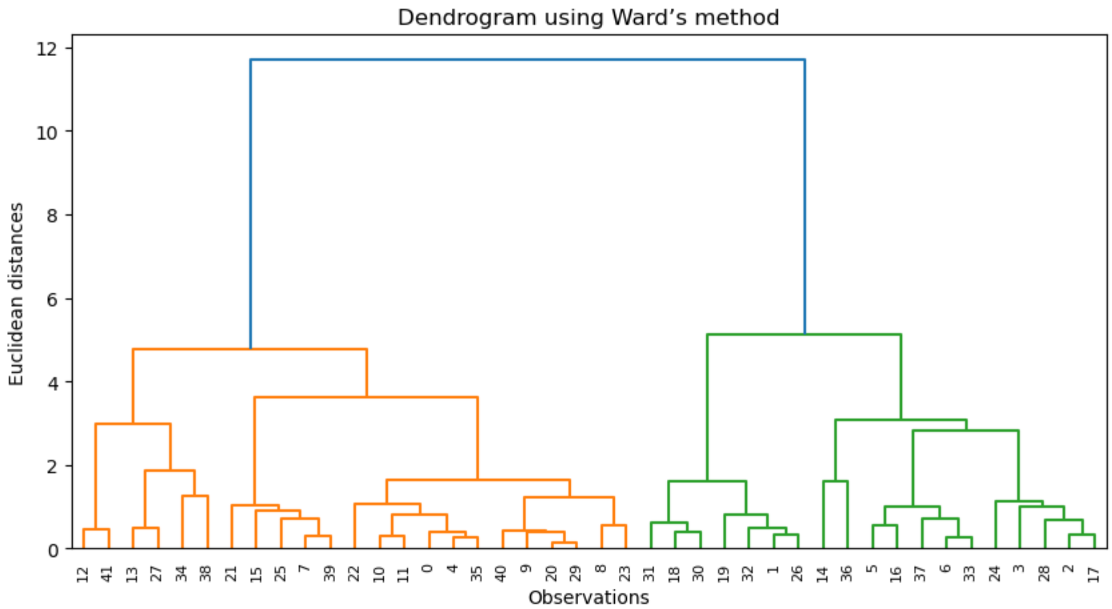
```
/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency
information was provided, so inferred frequency MS will be used.
  self._init_dates(dates, freq)
/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/statsmodels/tsa/holtwinters/model.py:915: ConvergenceWarning:
Optimization failed to converge. Check mle_retvals.
  warnings.warn(
/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency
information was provided, so inferred frequency MS will be used.
  self._init_dates(dates, freq)
/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/statsmodels/tsa/holtwinters/model.py:915: ConvergenceWarning:
Optimization failed to converge. Check mle_retvals.
  warnings.warn(
```




```
MSE - Holt's Model: 71802642.22
MSE - Holt-Winters Model: 16439804.83
Recommended Model: Holt-Winters
```

[40]:

```
import scipy.cluster.hierarchy as sch
df = pd.read_csv(get_file_path("ClusterAnalysis_2025.csv"))

X = df.iloc[:, :].values


scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


linked = sch.linkage(X_scaled, method='ward')

# Plot the dendrogram
plt.figure(figsize=(10, 5))
dendrogram = sch.dendrogram(linked)
plt.title('Dendrogram using Ward’s method')
plt.xlabel('Observations')
plt.ylabel('Euclidean distances')
plt.show()
```



[43]:
```
from sklearn.cluster import KMeans

k = 2

kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
df['Cluster'] = kmeans.fit_predict(X_scaled)


print("Cluster Centers:\n", kmeans.cluster_centers_)

plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=df['Cluster'], cmap='viridis',␣
  ↪alpha=0.7)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],␣
  ↪s=200, c='red', marker='X')  # Cluster centers
plt.title('K-Means Clustering')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```

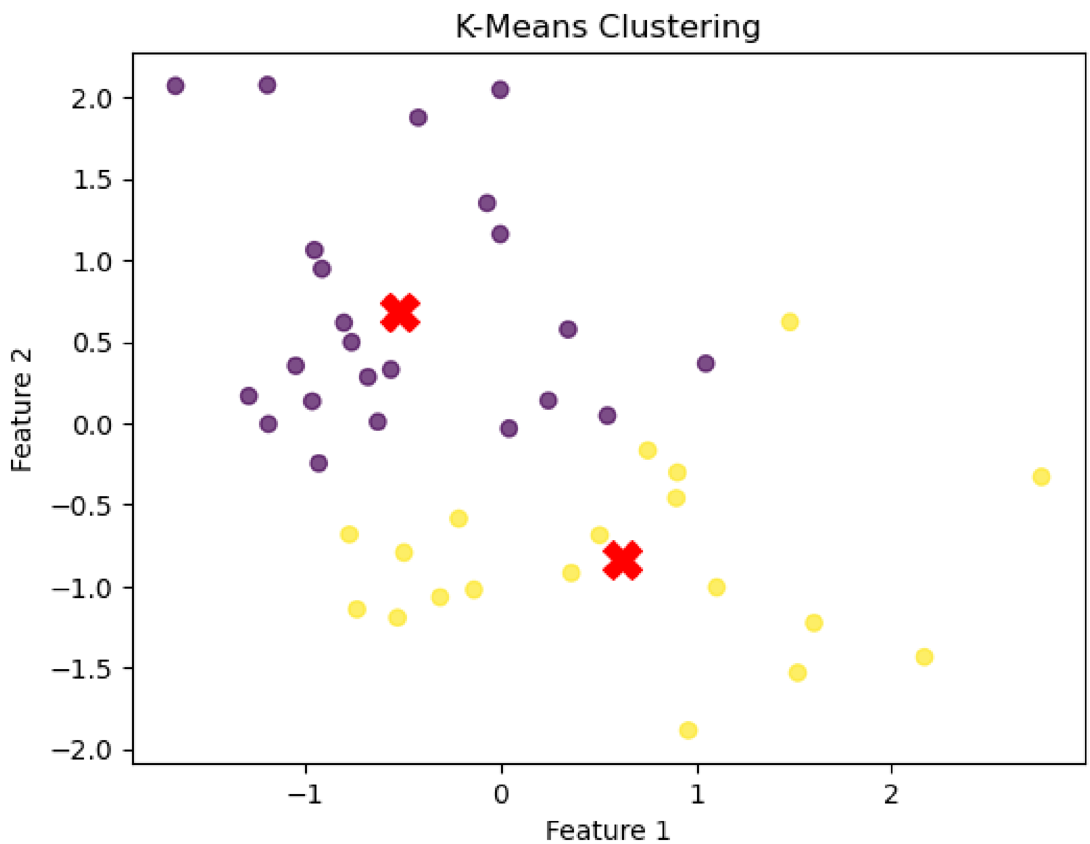
```
Exception ignored on calling ctypes callback function: <function _ThreadpoolInfo
._find_modules_with_dl_iterate_phdr.<locals>.match_module_callback at
0x7feb7770cca0>
Traceback (most recent call last):
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 400, in match_module_callback
    self._make_module_from_path(filepath)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
```

```
packages/threadpoolctl.py", line 515, in _make_module_from_path
    module = module_class(filepath, prefix, user_api, internal_api)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 606, in __init__
    self.version = self.get_version()
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 646, in get_version
    config = get_config().split()
AttributeError: 'NoneType' object has no attribute 'split'
Exception ignored on calling ctypes callback function: <function _ThreadpoolInfo
._find_modules_with_dl_iterate_phdr.<locals>.match_module_callback at
0x7feb7770cca0>
Traceback (most recent call last):
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 400, in match_module_callback
    self._make_module_from_path(filepath)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 515, in _make_module_from_path
    module = module_class(filepath, prefix, user_api, internal_api)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 606, in __init__
    self.version = self.get_version()
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 646, in get_version
    config = get_config().split()
AttributeError: 'NoneType' object has no attribute 'split'
Exception ignored on calling ctypes callback function: <function _ThreadpoolInfo
._find_modules_with_dl_iterate_phdr.<locals>.match_module_callback at
0x7feb7770cca0>
Traceback (most recent call last):
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 400, in match_module_callback
    self._make_module_from_path(filepath)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 515, in _make_module_from_path
    module = module_class(filepath, prefix, user_api, internal_api)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 606, in __init__
    self.version = self.get_version()
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 646, in get_version
    config = get_config().split()
AttributeError: 'NoneType' object has no attribute 'split'
Exception ignored on calling ctypes callback function: <function _ThreadpoolInfo
._find_modules_with_dl_iterate_phdr.<locals>.match_module_callback at
0x7feb7770cca0>
Traceback (most recent call last):
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
```

```
packages/threadpoolctl.py", line 400, in match_module_callback
    self._make_module_from_path(filepath)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 515, in _make_module_from_path
    module = module_class(filepath, prefix, user_api, internal_api)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 606, in __init__
    self.version = self.get_version()
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 646, in get_version
    config = get_config().split()
AttributeError: 'NoneType' object has no attribute 'split'
Exception ignored on calling ctypes callback function: <function _ThreadpoolInfo
._find_modules_with_dl_iterate_phdr.<locals>.match_module_callback at
0x7feb7770cca0>
Traceback (most recent call last):
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 400, in match_module_callback
    self._make_module_from_path(filepath)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 515, in _make_module_from_path
    module = module_class(filepath, prefix, user_api, internal_api)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 606, in __init__
    self.version = self.get_version()
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 646, in get_version
    config = get_config().split()
AttributeError: 'NoneType' object has no attribute 'split'
Exception ignored on calling ctypes callback function: <function _ThreadpoolInfo
._find_modules_with_dl_iterate_phdr.<locals>.match_module_callback at
0x7feb9d86d7e0>
Traceback (most recent call last):
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 400, in match_module_callback
    self._make_module_from_path(filepath)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 515, in _make_module_from_path
    module = module_class(filepath, prefix, user_api, internal_api)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 606, in __init__
    self.version = self.get_version()
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 646, in get_version
    config = get_config().split()
AttributeError: 'NoneType' object has no attribute 'split'
Exception ignored on calling ctypes callback function: <function _ThreadpoolInfo
._find_modules_with_dl_iterate_phdr.<locals>.match_module_callback at
```

```
0x7feb7770cca0>
Traceback (most recent call last):
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 400, in match_module_callback
    self._make_module_from_path(filepath)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 515, in _make_module_from_path
    module = module_class(filepath, prefix, user_api, internal_api)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 606, in __init__
    self.version = self.get_version()
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 646, in get_version
    config = get_config().split()
AttributeError: 'NoneType' object has no attribute 'split'
Exception ignored on calling ctypes callback function: <function _ThreadpoolInfo
._find_modules_with_dl_iterate_phdr.<locals>.match_module_callback at
0x7feb7770cca0>
Traceback (most recent call last):
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 400, in match_module_callback
    self._make_module_from_path(filepath)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 515, in _make_module_from_path
    module = module_class(filepath, prefix, user_api, internal_api)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 606, in __init__
    self.version = self.get_version()
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 646, in get_version
    config = get_config().split()
AttributeError: 'NoneType' object has no attribute 'split'
Exception ignored on calling ctypes callback function: <function _ThreadpoolInfo
._find_modules_with_dl_iterate_phdr.<locals>.match_module_callback at
0x7feb9d86d7e0>
Traceback (most recent call last):
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 400, in match_module_callback
    self._make_module_from_path(filepath)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 515, in _make_module_from_path
    module = module_class(filepath, prefix, user_api, internal_api)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 606, in __init__
    self.version = self.get_version()
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 646, in get_version
    config = get_config().split()
```

```
AttributeError: 'NoneType' object has no attribute 'split'
Exception ignored on calling ctypes callback function: <function _ThreadpoolInfo
._find_modules_with_dl_iterate_phdr.<locals>.match_module_callback at
0x7feb7770cca0>
Traceback (most recent call last):
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 400, in match_module_callback
    self._make_module_from_path(filepath)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 515, in _make_module_from_path
    module = module_class(filepath, prefix, user_api, internal_api)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 606, in __init__
    self.version = self.get_version()
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 646, in get_version
    config = get_config().split()
AttributeError: 'NoneType' object has no attribute 'split'
Exception ignored on calling ctypes callback function: <function _ThreadpoolInfo
._find_modules_with_dl_iterate_phdr.<locals>.match_module_callback at
0x7feb7770cca0>
Traceback (most recent call last):
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 400, in match_module_callback
    self._make_module_from_path(filepath)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 515, in _make_module_from_path
    module = module_class(filepath, prefix, user_api, internal_api)
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 606, in __init__
    self.version = self.get_version()
  File "/home/musiliandrew/anaconda3/lib/python3.10/site-
packages/threadpoolctl.py", line 646, in get_version
    config = get_config().split()
AttributeError: 'NoneType' object has no attribute 'split'

Cluster Centers:
 [[-0.51458263  0.68771988  0.78079648]
 [ 0.62291581 -0.83250301 -0.94517468]]
```



[44]:
```
from scipy.spatial.distance import cdist
from sklearn.metrics import pairwise_distances

def dunn_index(X, labels):
    unique_clusters = np.unique(labels)
    intra_cluster_distances = []
    inter_cluster_distances = []

    # Compute intra-cluster distances (maximum distance within each cluster)
    for cluster in unique_clusters:
        points = X[labels == cluster]
        if len(points) > 1:
            intra_cluster_distances.append(np.max(pairwise_distances(points)))

    # Compute inter-cluster distances (minimum distance between clusters)
    for i in range(len(unique_clusters)):
        for j in range(i + 1, len(unique_clusters)):
            points_i = X[labels == unique_clusters[i]]
            points_j = X[labels == unique_clusters[j]]
            inter_cluster_distances.append(np.min(cdist(points_i, points_j)))
```

```
    return np.min(inter_cluster_distances) / np.max(intra_cluster_distances)

# Compute Dunn Index
dunn_value = dunn_index(X_scaled, kmeans.labels_)
print("Dunn Index:", dunn_value)
```

Dunn Index: 0.2315306889122367

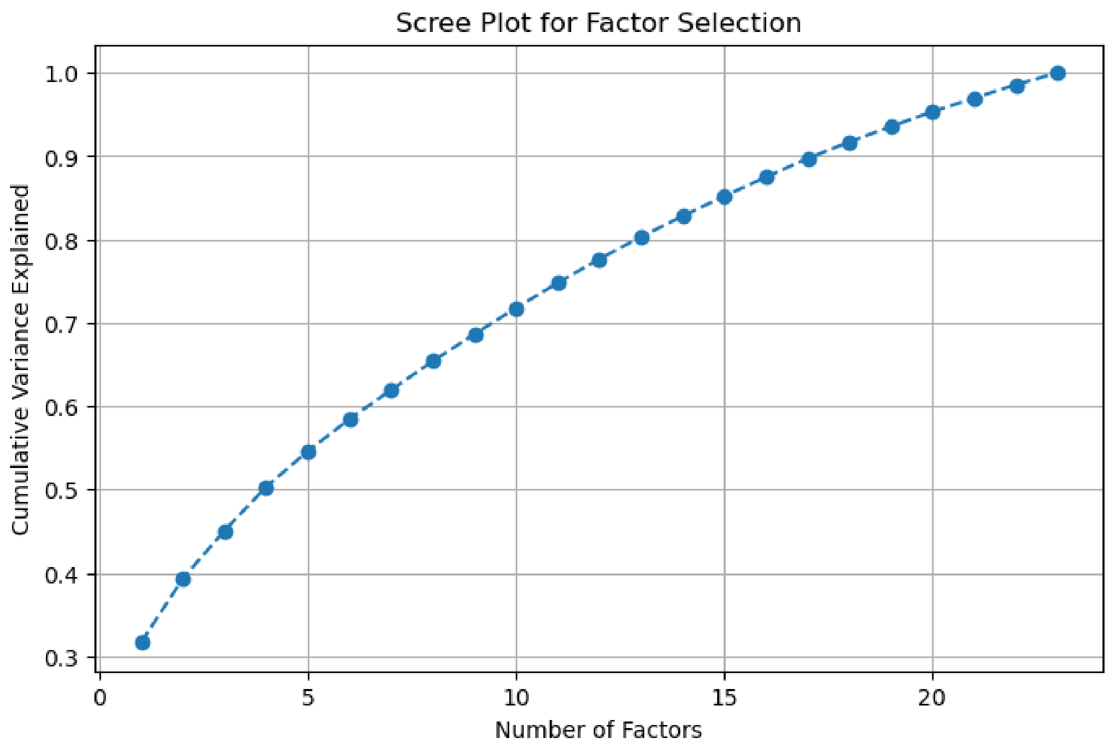
[ ]:

```

```

[46]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from factor_analyzer import FactorAnalyzer

# Load dataset
df = pd.read_csv(get_file_path("2025_FactorAnalysis_Dataset.csv"))  # Replace␣
 ↪with actual file

# Step 1: Check for missing values and handle them
df.dropna(inplace=True)  # Dropping missing values for simplicity

# Step 2: Standardize the data (PCA is affected by scale)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df)

# Step 3: Determine the number of factors using PCA
pca = PCA()
pca.fit(df_scaled)
explained_variance = pca.explained_variance_ratio_

# Plot Scree plot
plt.figure(figsize=(8,5))
plt.plot(range(1, len(explained_variance) + 1), explained_variance.cumsum(),␣
 ↪marker='o', linestyle='--')
plt.xlabel("Number of Factors")
plt.ylabel("Cumulative Variance Explained")
plt.title("Scree Plot for Factor Selection")
plt.grid()
plt.show()

# Select factors where cumulative variance reaches ~70% (common threshold)
n_factors = np.argmax(explained_variance.cumsum() >= 0.70) + 1
print(f"Selected Number of Factors: {n_factors}")
```

```python
# Step 4: Compute the Cumulative Variance for Selected Factors
cumulative_variance = explained_variance[:n_factors].sum()
print(f"Cumulative Variance Explained by Selected Factors: {cumulative_variance:
↪.2%}")

# Step 5: Perform PCA with Varimax Rotation
fa = FactorAnalyzer(n_factors, rotation="varimax")
fa.fit(df_scaled)
factor_loadings = fa.loadings_

# Step 6: Interpretation - Print Factor Loadings
factor_loadings_df = pd.DataFrame(factor_loadings, index=df.columns,␣
↪columns=[f'Factor {i+1}' for i in range(n_factors)])
print("\nFactor Loadings after Varimax Rotation:")
print(factor_loadings_df)

# Identify variables highly associated with each factor
for i in range(n_factors):
    associated_vars = factor_loadings_df.iloc[:, i].abs().
↪sort_values(ascending=False).index[:3]  # Top 3 variables per factor
    print(f"\nTop Variables Associated with Factor {i+1}:␣
↪{list(associated_vars)}")
```



```
Selected Number of Factors: 10
Cumulative Variance Explained by Selected Factors: 71.81%

Factor Loadings after Varimax Rotation:
     Factor 1  Factor 2  Factor 3  Factor 4  Factor 5  Factor 6  Factor 7  \
q01  0.177639  0.581011  0.244872 -0.071363  0.096373  0.021116 -0.010911
q02 -0.047620 -0.079967 -0.011805  0.459670 -0.149186 -0.050072  0.065653
q03 -0.178200 -0.327755 -0.185804  0.494354 -0.213854 -0.066752 -0.010930
q04  0.236886  0.557873  0.229563 -0.103305  0.138406  0.093506  0.009499
q05  0.232120  0.521850  0.160935 -0.085474  0.121079  0.058713 -0.050905
q06  0.754655  0.084064  0.129639 -0.057663  0.006659  0.099887 -0.054098
q07  0.575663  0.266667  0.179769 -0.144333  0.163788  0.107179 -0.034660
q08  0.126021  0.209417  0.764718  0.002217  0.080449  0.062487 -0.027782
q09 -0.089330 -0.062031  0.051621  0.620680 -0.032541 -0.057504  0.040708
q10  0.348726  0.208467  0.103830 -0.127250 -0.019253  0.129090 -0.033906
q11  0.256313  0.186981  0.721658 -0.165682  0.130074  0.082082 -0.034295
q12  0.380139  0.379269  0.139529 -0.200534  0.227174  0.086131 -0.015991
q13  0.575100  0.223219  0.252366 -0.132139  0.102761  0.051763 -0.058449
q14  0.453716  0.279604  0.171203 -0.147544  0.164795  0.127506 -0.044892
q15  0.286246  0.169792  0.201814 -0.161065  0.090251  0.895901 -0.039504
q16  0.207664  0.527495  0.200616 -0.199580  0.144719  0.214604 -0.024059
q17  0.238965  0.253096  0.642162 -0.029374  0.095279  0.109221 -0.039983
q18  0.616617  0.242376  0.181302 -0.160264  0.152936  0.038816 -0.019938
q19 -0.160046 -0.121016 -0.087396  0.366953 -0.183599 -0.044411  0.092367
q20  0.050055  0.138548  0.114359 -0.213396  0.671744  0.050285 -0.001281
q21  0.299289  0.304111  0.190421 -0.170676  0.517605  0.045351 -0.011784
q22 -0.095317 -0.052774 -0.046622  0.332056 -0.020735 -0.039312  0.931972
q23 -0.044307  0.025509 -0.077634  0.264774  0.019961  0.004137  0.150833

     Factor 8  Factor 9  Factor 10
q01  0.155748  0.060399  -0.063942
q02  0.034674 -0.074374  -0.004886
q03 -0.054201 -0.105854  -0.166899
q04 -0.068970  0.077279   0.130515
q05 -0.017671 -0.035600  -0.003033
q06 -0.047245 -0.115861   0.007996
q07  0.006246 -0.041872   0.307867
q08  0.020844 -0.003190   0.057902
q09 -0.001796  0.013057   0.116626
q10  0.057729 -0.054041  -0.114570
q11  0.002661  0.011878  -0.101274
q12  0.004851  0.312412   0.055931
q13  0.034037  0.288048  -0.123808
q14  0.127789  0.167757   0.078584
q15  0.064579  0.029242   0.017105
q16  0.487123  0.018530   0.005668
```

```
q17  0.094310  0.081344   0.095827
q18  0.127589  0.176517   0.063069
q19 -0.078742 -0.039607  -0.070734
q20  0.013416  0.015427  -0.048586
q21  0.077138  0.070395   0.207623
q22  0.003813 -0.025029   0.012580
q23 -0.054655  0.052620  -0.056677

Top Variables Associated with Factor 1: ['q06', 'q18', 'q07']

Top Variables Associated with Factor 2: ['q01', 'q04', 'q16']

Top Variables Associated with Factor 3: ['q08', 'q11', 'q17']

Top Variables Associated with Factor 4: ['q09', 'q03', 'q02']

Top Variables Associated with Factor 5: ['q20', 'q21', 'q12']

Top Variables Associated with Factor 6: ['q15', 'q16', 'q10']

Top Variables Associated with Factor 7: ['q22', 'q23', 'q19']

Top Variables Associated with Factor 8: ['q16', 'q01', 'q14']

Top Variables Associated with Factor 9: ['q12', 'q13', 'q18']

Top Variables Associated with Factor 10: ['q07', 'q21', 'q03']
```