COS40007 Artificial Intelligence for Engineering

Portfolio Assessment-1: "Hello Machine Learning for Engineering"

Name: Arnob Ghosh

Student Number: 103494114

Studio Class:

Dataset Information

Dataset Selected: Combined Cycle Power Plant dataset

Reason for Choice:

My area of interest as a software engineering major is power plant performance and efficiency. This dataset offers insightful information about how various ambient factors affect power output, which is directly relevant to my area of study.

Summary of Exploratory Data Analysis (EDA):

9568 data points gathered over a six-year period (2006-2011) from a Combined Cycle Power Plant are included in the collection. The target variable is the plant's net hourly electrical energy output (PE), and the features include hourly average ambient variables like temperature (AT), ambient pressure (AP), relative humidity (RH), and exhaust vacuum (V).

Key Insights from EDA:

1. Correlation Matrix:

- There is a significant negative association (-0.95) between temperature (AT) and PE.
- There is a high negative association (-0.87) between Exhaust Vacuum (V) and PE.
- There are moderate associations between PE and relative humidity (RH) and ambient pressure (AP) (0.52 and -0.39, respectively).

2. Distribution of Features:

- The distributions of temperature (AT) and net electrical output (PE) are essentially normal.
- The distributions of relative humidity (RH) and exhaust vacuum (V) are slightly skewed.

3. Data Visualizations:

Visual confirmation of the relationships between the variables was provided via feature distribution plots and the correlation matrix.

Class Labeling for Target Variable / Developing Ground Truth Data:

The target variable (PE) was divided into three class labels based on its distribution:

- Low Output: PE < 440 MW
- Medium Output: 440 MW ≤ PE < 470 MW
- High Output: PE ≥ 470 MW

Class Distribution:

- Low: 2905
- Medium: 4238
- High: 2425

Feature Engineering and Feature Selection:

Normalization:

Using StandardScaler, numerical features (AT, V, AP, and RH) were standardized to have a mean of 0 and a standard deviation of 1.

Composite Features:

Two composite features were created:

- AT_AP: Product of AT and AP.
- V_RH: Product of V and RH.

These characteristics were chosen because they may offer more information about the operation of the power plant.

Training and Decision Tree Model Development:

Eighty percent of the dataset was used for training and twenty percent for testing. Various feature sets were used to train decision tree models.

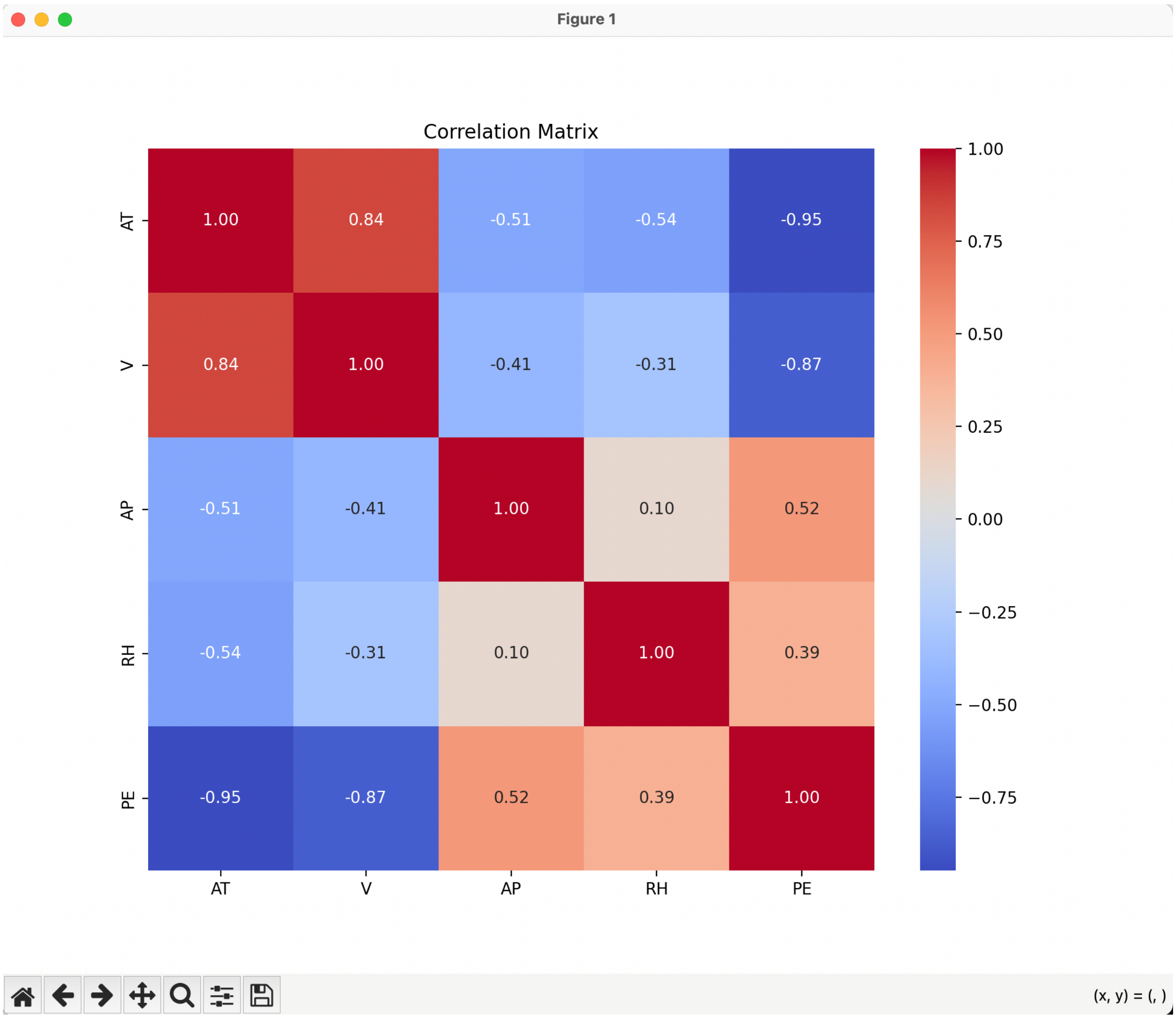
Feature Sets:

1. Set 1: AT, V, AP, RH
2. Set 2: AT, V, AP, RH, AT_AP
3. Set 3: AT, V, AP, RH, V_RH
4. Set 4: AT, V, AP, RH, AT_AP, V_RH
5. Set 5: AT_AP, V_RH

Feature Set	Accuracy
Set 2	86.99%
Set 4	86.73%
Set 1	86.68%
Set 3	86.21%
Set 5	58.62%

Brief Summary of Observations in the Comparison Table:

- Set 2 (AT, V, AP, RH, AT_AP) attained the maximum accuracy, suggesting that AT_AP, a composite feature, greatly enhances the performance of the model.
- Set 4 (AT, V, AP, RH, AT_AP, V_RH) likewise gave good results, demonstrating that the value added by the two composite characteristics is comparable to that of employing AT_AP alone.
- Set 1 (AT, V, AP, RH) offered a strong foundational performance.
- Set 3 (AT, V, AP, RH, V_RH) shown that AT_AP adds more value than the V_RH composite feature alone.
- Set 5 (AT_AP, V_RH) produced the lowest accuracy, indicating that using the original features is more successful than relying only on composite features.




```
"/Users/ag47/Documents/Swinburne Class/AI for Engineering/Week 2/Week 2 Portfolio Task /pythonProject/.venv/bin/python" /Users/ag47/Documents/Swinburne Class/AI
First few rows of the dataset:
      AT      V       AP     RH      PE
0  14.96  41.76  1024.07  73.17  463.26
1  25.18  62.96  1020.04  59.08  444.37
2   5.11  39.40  1012.16  92.14  488.56
3  20.86  57.32  1010.24  76.64  446.48
4  10.82  37.50  1009.23  96.62  473.90

Summary statistics of the dataset:
                AT            V           AP           RH           PE
count  9568.000000  9568.000000  9568.000000  9568.000000  9568.000000
mean     19.651231    54.305804  1013.259078    73.308978   454.365009
std       7.452473    12.707893     5.938784    14.600269    17.066995
min       1.810000    25.360000   992.890000    25.560000   420.260000
25%      13.510000    41.740000  1009.100000    63.327500   439.750000
50%      20.345000    52.080000  1012.940000    74.975000   451.550000
75%      25.720000    66.540000  1017.260000    84.830000   468.430000
max      37.110000    81.560000  1033.300000   100.160000   495.760000

Correlation matrix of the dataset:
          AT         V        AP        RH        PE
AT  1.000000  0.844107 -0.507549 -0.542535 -0.948128
V   0.844107  1.000000 -0.413502 -0.312187 -0.869780
AP -0.507549 -0.413502  1.000000  0.099574  0.518429
RH -0.542535 -0.312187  0.099574  1.000000  0.389794
PE -0.948128 -0.869780  0.518429  0.389794  1.000000
```

Source Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Load the dataset
file_path = '********';
data = pd.read_excel(file_path)

# Display the first few rows of the dataset
print("First few rows of the dataset:")
print(data.head())

# Summary statistics
print("\nSummary statistics of the dataset:")
summary_stats = data.describe()
print(summary_stats)

# Correlation matrix
print("\nCorrelation matrix of the dataset:")
correlation_matrix = data.corr()
print(correlation_matrix)

# Visualize the correlation matrix
```

```
plt.figure(figsize=(10, 8))

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt='.2f')

plt.title('Correlation Matrix')

plt.show()


# Visualize the distribution of features

fig, axs = plt.subplots(2, 3, figsize=(15, 10))

sns.histplot(data['AT'], kde=True, ax=axs[0,
0]).set(title='Temperature (AT)')

sns.histplot(data['V'], kde=True, ax=axs[0, 1]).set(title='Exhaust
Vacuum (V)')

sns.histplot(data['AP'], kde=True, ax=axs[0, 2]).set(title='Ambient
Pressure (AP)')

sns.histplot(data['RH'], kde=True, ax=axs[1, 0]).set(title='Relative
Humidity (RH)')

sns.histplot(data['PE'], kde=True, ax=axs[1, 1]).set(title='Net
Electrical Output (PE)')

fig.delaxes(axs[1, 2])

plt.tight_layout()

plt.show()


# Class Label Definition

data['PE_class'] = pd.cut(data['PE'], bins=[data['PE'].min(), 440,
470, data['PE'].max()], labels=['Low', 'Medium', 'High'])

print("\nClass distribution in PE_class:")

print(data['PE_class'].value_counts())


# Normalization of numerical features

scaler = StandardScaler()
```

```
data[['AT', 'V', 'AP', 'RH']] = scaler.fit_transform(data[['AT',
'V', 'AP', 'RH']])
print("\nFirst few rows after normalization:")
print(data.head())

# Feature Engineering: Creating composite features
data['AT_AP'] = data['AT'] * data['AP']
data['V_RH'] = data['V'] * data['RH']
print("\nFirst few rows after feature engineering:")
print(data.head())

# Check for missing values in the dataset
missing_values = data.isnull().sum()
print("\nMissing values in the dataset:")
print(missing_values)

# Drop the row with missing value in PE_class
data_cleaned = data.dropna(subset=['PE_class'])
print("\nMissing values after cleaning:")
print(data_cleaned.isnull().sum())

# Split the cleaned data into training and testing sets
X_cleaned = data_cleaned.drop(columns=['PE', 'PE_class'])
y_cleaned = data_cleaned['PE_class']
X_train_cleaned, X_test_cleaned, y_train_cleaned, y_test_cleaned =
train_test_split(X_cleaned, y_cleaned, test_size=0.2,
random_state=42)

# Define the feature sets
feature_sets = {
```

```
    'Set 1': ['AT', 'V', 'AP', 'RH'],
    'Set 2': ['AT', 'V', 'AP', 'RH', 'AT_AP'],
    'Set 3': ['AT', 'V', 'AP', 'RH', 'V_RH'],
    'Set 4': ['AT', 'V', 'AP', 'RH', 'AT_AP', 'V_RH'],
    'Set 5': ['AT_AP', 'V_RH']
}

# Dictionary to store accuracy results for cleaned data
accuracy_results_cleaned = {}

# Train and evaluate models for each feature set on cleaned data
for set_name, features in feature_sets.items():
    # Extract the features for the current set
    X_train_set = X_train_cleaned[features]
    X_test_set = X_test_cleaned[features]

    # Initialize and train the decision tree classifier
    clf = DecisionTreeClassifier(random_state=42)
    clf.fit(X_train_set, y_train_cleaned)

    # Predict on the test set
    y_pred = clf.predict(X_test_set)

    # Calculate accuracy
    accuracy = accuracy_score(y_test_cleaned, y_pred)
    accuracy_results_cleaned[set_name] = accuracy

# Convert accuracy results to a DataFrame for easy comparison
```

```
accuracy_df_cleaned =
pd.DataFrame.from_dict(accuracy_results_cleaned, orient
```