Machine Learning

Facilitator: Dr Amit Swamy

Group-9

Rohith	23WU0102103
Keerthan	23WU0102232
Jatin	23WU0102116
Akhil	23WU0102094
Muktesh	23WU0102157

Report-1

Ranking

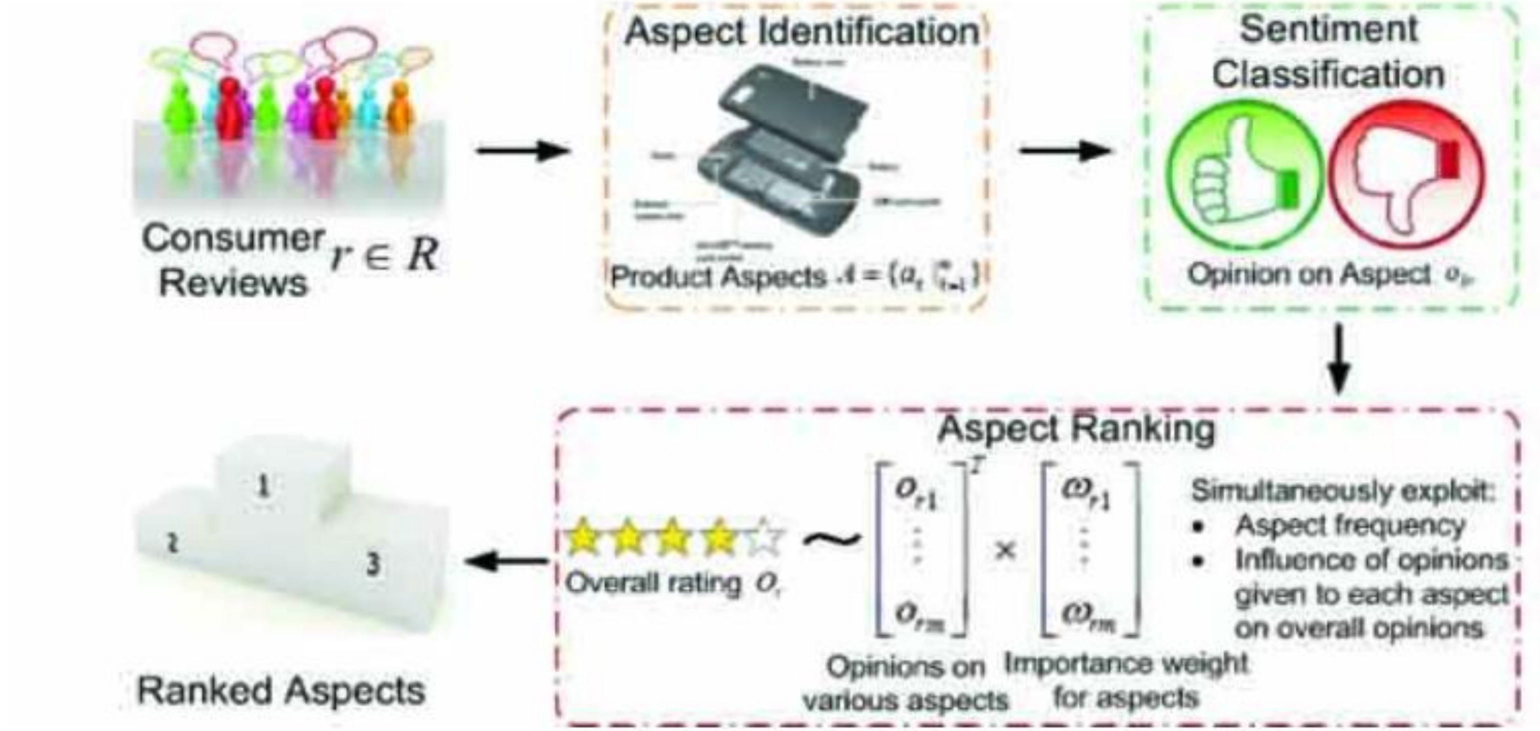
- “Ranking" commonly refers to extracting a scoring approach from statistics using algorithms.
- The process of placing objects, entities, or pieces in a certain order to represent their relative significance, importance, or worth in a particular context is known as Ranking.

Product Aspect Ranking and Its Applications

Objective: A product aspect ranking framework presented in the paper helps identify vital product elements through consumer reviews to achieve better usability along with better knowledge retrieval.

Approach: The identification of key aspects depends on aspect frequency and opinion influence using dependency parsing, sentiment analysis, and probabilistic aspect ranking computations.

Results & Applications: The framework demonstrated its effectiveness through experimental results applied to 21 products spanning eight different domains. The framework demonstrates its practical benefits through improved document sentiment detection and review synthesis capabilities.


Consumer Reviews $r \in R$
Aspect Identification
Product Aspects
Sentiment Classification
Opinion on Aspect
Aspect Ranking
Overall rating O_r
Simultaneously exploit:
• Aspect frequency
• Influence of opinions given to each aspect on overall opinions
Opinions on various aspects
Importance weight for aspects
Ranked Aspects

Clustering

- The task of grouping data points based on their similarity is called Clustering or Cluster Analysis.
- This method is defined under the branch of Unsupervised learning, which aims at gaining insights from Unlabelled data points, that is, unlike supervised learning we don’t have a target variable.

Validity-guided (re)clustering with applications to image segmentation

•Objective: VGC presents a Validity-Guided (Re)Clustering (VGC) algorithm that enhances image segmentation through cluster-validity assessments to overcome clustering algorithm constraints.

•Approach: VGC uses an iterative split-and-merge process to enhance the initial fuzzy clustering partition while keeping modifications that enhance partition validity.

•Results & Applications: The VGC algorithm shows superior results over fuzzy c-means in synthetic and real-world data applications specifically MRI segmentation where it matches the effectiveness of supervised k-nearest-neighbors according to radiological validation.
.

Dimensionality Reduction

- The number of input features, variables, or columns present in a given dataset is known as dimensionality, and the process to reduce these features is called dimensionality reduction.
- A dataset contains a huge number of input features in various cases, which makes the predictive modeling task more complicated.
- It is very difficult to visualize or make predictions for the training dataset with a high number of features, for such cases, dimensionality reduction techniques are required.

Performance Evaluation of Dimensionality Reduction Techniques on High-Dimensional Data

•Objective: This paper analyzes three common dimension reduction techniques: Independent Component Analysis (ICA) alongside Principal Component Analysis (PCA) and Non-Negative Matrix Factorization (NMF) as solutions to handle high-dimensional datasets.

•Approach: This research applies standardized benchmarks to check real-world datasets to measure the performance efficiency and effectiveness of the method.

•Conclusion: Data scientists gain direction from this study regarding which dimensionality reduction method to choose by considering effectiveness alongside computational efficiency.

Citations

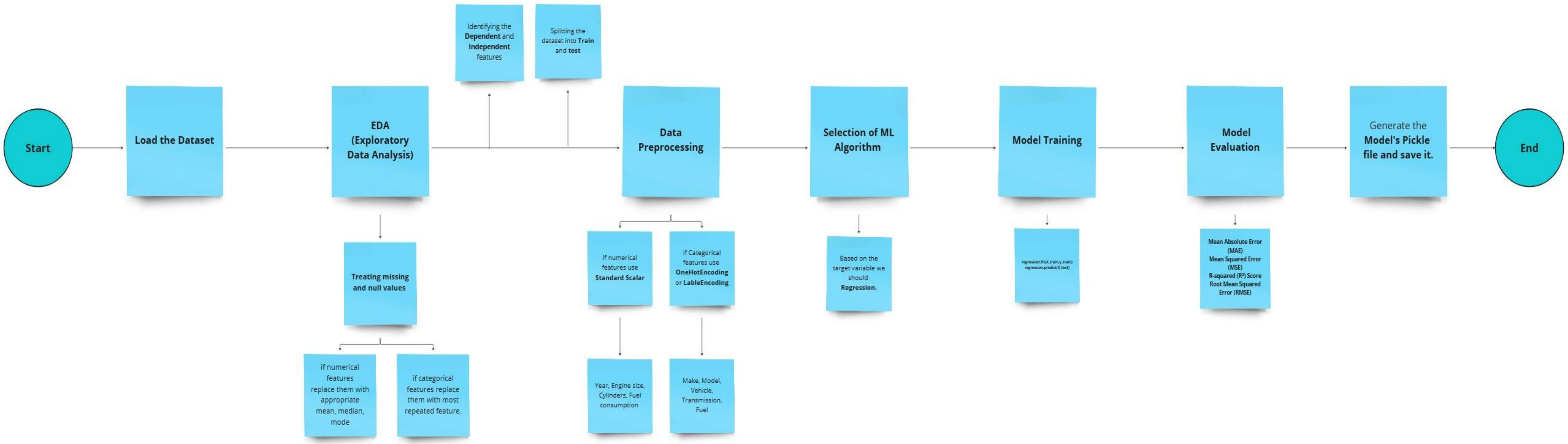
- Zha, Z.-J., Yu, J., Tang, J., Wang, M., & Chua, T.-S. (2014). Product aspect ranking and its applications. *IEEE Transactions on Knowledge and Data Engineering*, *26*(5), 1211-1224. https://doi.org/10.1109/TKDE.2013.136
- Bensaid, A. M., et al. (1996). Validity-guided (re)clustering with applications to image segmentation. *IEEE Transactions on Fuzzy Systems*, *4*(2), 112-123. https://doi.org/10.1109/91.493905
- Vikram, M., Pavan, R., Dineshbhai, N. D., & Mohan, B. (2019). Performance evaluation of dimensionality reduction techniques on high dimensional data. In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)* (pp. 1169-1174). IEEE. https://doi.org/10.1109/ICOEI.2019.8862526

Report-2

Flowchart for the solution

Task-2

Dataset-2 (Predicting fuel consumption based on vehicle features)



Report-3

Competition

Task-3

```python
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

[18] Python

```python
df = pd.read_csv('play_tennis.csv')
df.drop(["day"],axis=1,inplace=True)
```

[19] Python

```python
df
```

[20] Python

	outlook	temp	humidity	wind	play
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes
13	Rain	Mild	High	Strong	No

Task-3

Report-3
Competition

```
[21] df["humidity"].value_counts()
```

```
humidity
High      7
Normal    7
Name: count, dtype: int64
```

```
[22] df["outlook"].value_counts()
```

```
outlook
Sunny       5
Rain        5
Overcast    4
Name: count, dtype: int64
```

```
[23] df["temp"].value_counts()
```

```
temp
Mild    6
Hot     4
Cool    4
Name: count, dtype: int64
```

```
[24] df["wind"].value_counts()
```

```
wind
Weak      8
Strong    6
Name: count, dtype: int64
```

```
[25] df.dtypes
```

```
outlook     object
temp        object
humidity    object
wind        object
play        object
```

Report-3

Task-3

Competition

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print(confusion_matrix(y_pred, y_test))
print(accuracy_score(y_pred, y_test))
print(classification_report(y_pred, y_test))
```

[31] ✓ 0.0s Python

```
[[0 0]
 [1 3]]
0.75
              precision    recall  f1-score   support

          No       0.00      0.00      0.00         0
         Yes       1.00      0.75      0.86         4

    accuracy                           0.75         4
   macro avg       0.50      0.38      0.43         4
weighted avg       1.00      0.75      0.86         4
```

```
import pickle
pickle.dump(Bnb,open('bnb.pkl','wb'))
```

[32] ✓ 0.0s Python

```
X_sample_data = pd.DataFrame([['Sunny','Cool','High','Strong']], columns=['outlook', 'temp', 'humidity', 'wind'])
X_sample_data
```

[35] ✓ 0.0s Python

	outlook	temp	humidity	wind
0	Sunny	Cool	High	Strong

```
X_sample = preprocessor.transform(X_sample_data)
y_sample_result = Bnb.predict(X_sample)
y_sample_result
```

[36] ✓ 0.0s Python

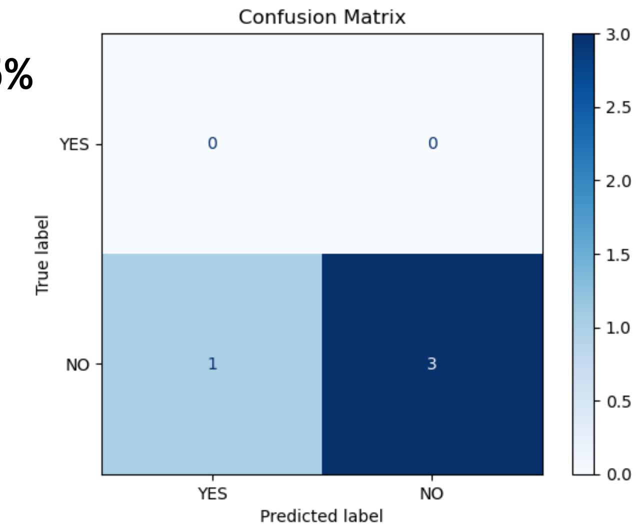
```
array(['No'], dtype='<U3')
```

Report-3

Competition

Task-3

Accuracy: 75%



Output:

```python
X_sample_data = pd.DataFrame([['Sunny','Cool','High','Strong']], columns=['outlook', 'temp', 'humidity', 'wind'])
X_sample_data
```

✓ 0.0s Python

	outlook	temp	humidity	wind
0	Sunny	Cool	High	Strong

```python
X_sample = preprocessor.transform(X_sample_data)
y_sample_result = Bnb.predict(X_sample)
y_sample_result
```

✓ 0.0s Python

```
array(['No'], dtype='<U3')
```

Report-4

Task-4

Competition Goal

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
import numpy as np
import matplotlib.pyplot as plt

X = np.array(X).reshape(-1, 1)
y = np.array(y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

poly = PolynomialFeatures(degree=2)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

model = LinearRegression()
model.fit(X_train_poly, y_train)

y_test_pred = model.predict(X_test_poly)

test_mse = mean_squared_error(y_test, y_test_pred)
test_rmse = np.sqrt(test_mse)
test_mae = mean_absolute_error(y_test, y_test_pred)
test_r2 = r2_score(y_test, y_test_pred)

print(f" Mean Squared Error: {test_mse}")
print(f" Mean Absolute Error: {test_mae}")
print(f" Root Mean Absolute Error: {test_rmse}")
print(f" R-squared: {test_r2}")

plt.scatter(X_train, y_train, label="Training Data", color="green", alpha=0.8)
plt.scatter(X_test, y_test, label="Testing Data", color="orange", alpha=0.8)
X_plot = np.linspace(X.min(), X.max(), 100).reshape(-1, 1)
y_plot = model.predict(poly.transform(X_plot))
plt.plot(X_plot, y_plot, color="black", label=f"Polynomial Fit (degree {2})")
plt.title("Polynomial Regression Fit with Train-Test Split")
plt.xlabel("Tractor Age")
plt.ylabel("Maintenance Cost")
plt.legend()
plt.show()
```

✓ 0.2s

Python

Report-4

Task-4

Competition Goal

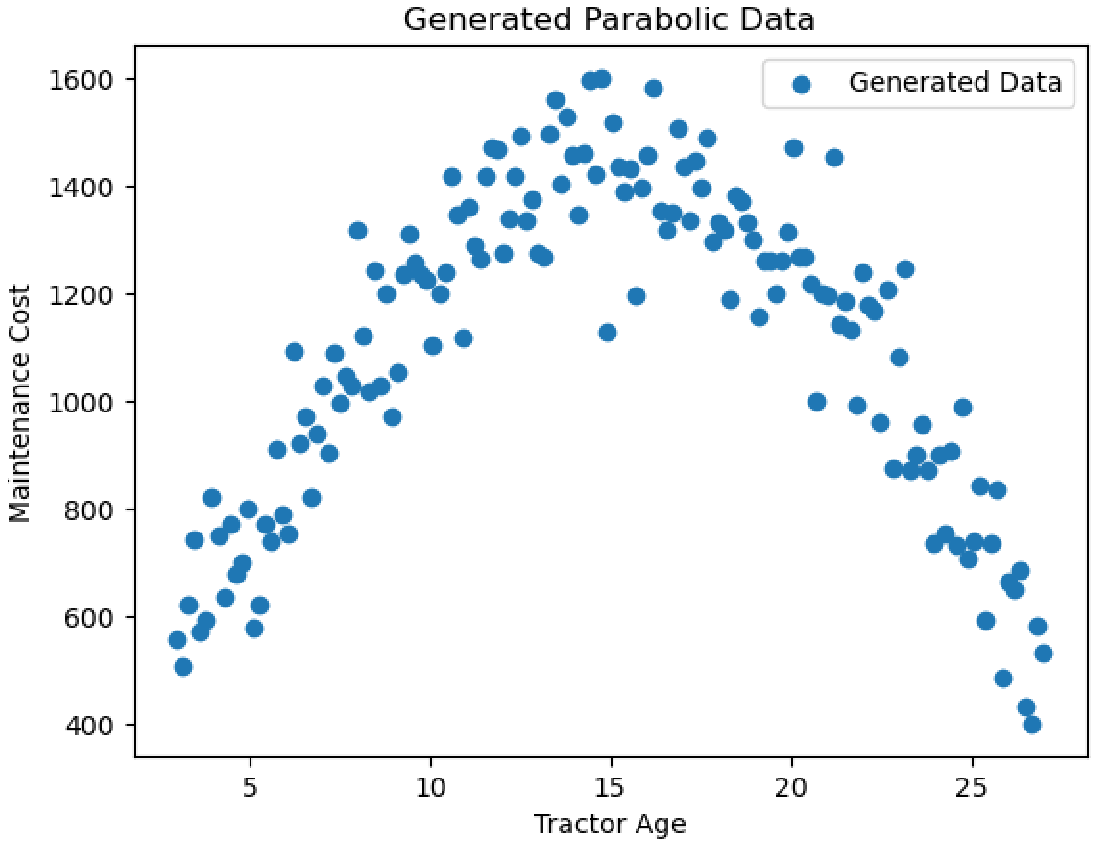
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Set seed for reproducibility
np.random.seed(42)

# Generate x values within the range seen in the image
x_parabolic = np.linspace(3, 27, 150)

# Define the parabolic function y = ax^2 + bx + c
a, b, c = -20, 600, 3000  # Coefficients approximating the trend in the image
noise = np.random.normal(0, 350, size=x_parabolic.shape)  # Adding noise

# Compute y values
y_parabolic = a * x_parabolic**2 + b * x_parabolic + c + noise

# Scale y values to range from 400 to 1600
y_min, y_max = 400, 1600
y_parabolic = (y_parabolic - np.min(y_parabolic)) / (np.max(y_parabolic) - np.min(y_parabolic))  # Normalize to 0-1
y_parabolic = y_parabolic * (y_max - y_min) + y_min  # Scale to range [400, 1600]

# Plot the generated dataset to ensure it resembles the uploaded image
plt.scatter(x_parabolic, y_parabolic, label="Generated Data")
plt.title("Generated Parabolic Data")
plt.xlabel("Tractor Age")
plt.ylabel("Maintenance Cost")
plt.legend()
plt.show()

# Prepare X and Y arrays for use
X = x_parabolic.tolist()
y = y_parabolic.tolist()
```

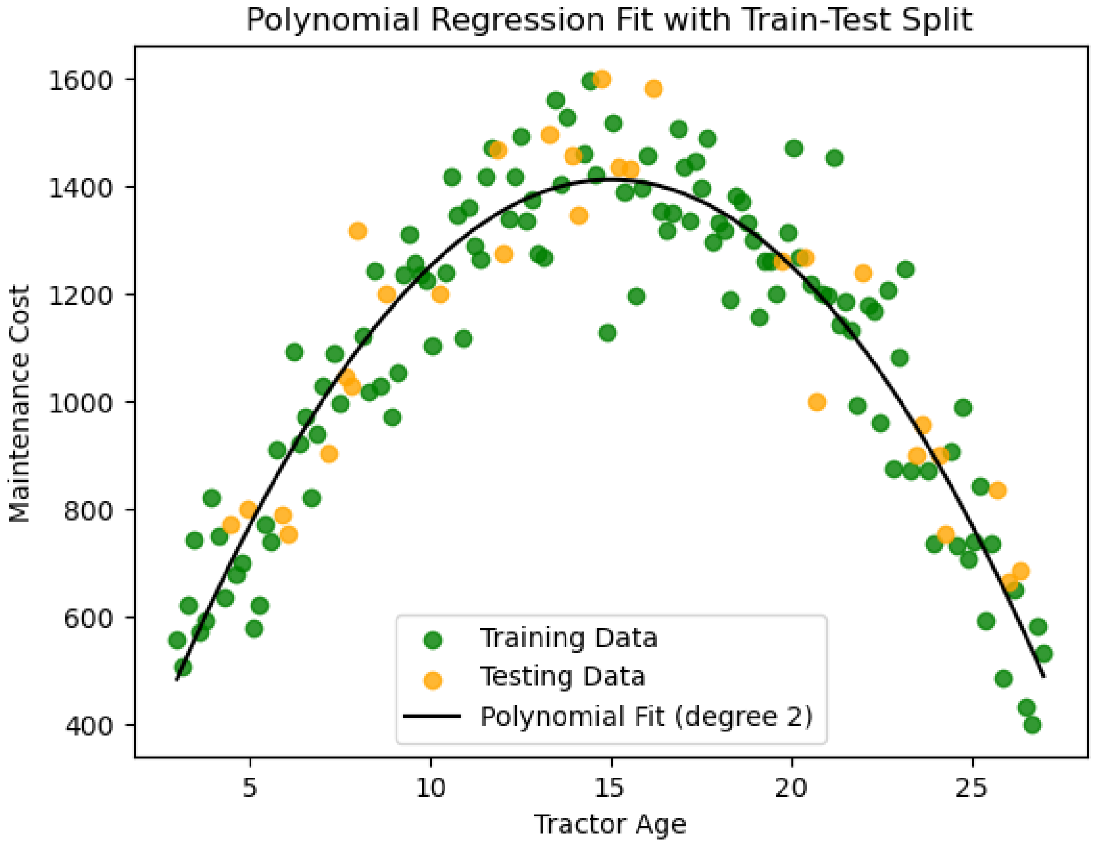
[3] ✓ 0.1s Python

Report-4

Task-4

Competition Goal





Results:-

```
Mean Squared Error: 10937.089362129414
Mean Absolute Error: 85.77776295657462
Root Mean Absolute Error: 104.58054007380825
R-squared: 0.8674798174597006
```