Project Phase 5

Github Repo: https://github.com/hemanthreddy-1711/Data-Mining-Project-Unstoppable

1. **Our Team Name and Members of The Team**

Name of the team: Unstoppable

Team Members:

1. **Hemanth Reddy Reddy Battula - hredd2@unh.newhaven.edu**
2. **Kola Avinash – akola@unh.newhaven.edu**
3. **Prem Niranjan Undavali – piunda1@unh.newhaven.edu**

2. **Our Research Question**

Research Question

How does the power demand of electric vehicle (EV) charging stations vary

across different cities and what regional factors influence these variations?

About the data set: https://www.kaggle.com/datasets/omarsobhy14/supercharge-locations

Our dataset has the details of street, city, zip, country, GPS, Kilo watts, Elev. It is collected from more than 5000 sessions from 100 drivers of 25 EV stations. It has data types which include string, time stamps, categorical, integer and more.

3. **List of Data mining techniques used**

We performed the data mining techniques below

a. Decision Tree
b. Random forest
c. Support vector Machine
d. Linear Regression
e. Naïve Bayes
f. Neural networks
g. Gradient Boosting

4. **Parameters and hyperparameters for the models that we have used**
    a. **Decision Tree**
        i. We have used random state = 50
        ii. Max_depth = 6
        iii. Min sample split =3
        iv. Min sample leaf = 3
    b. **Gradient Boosting**
        i. Random state = 42
        ii. n_estimators =100
        iii. learning_rate = 0.1
        iv. max_depth = 5
    c. **Random Forest**
        i. n_estimators = 100
        ii. max_depth =
        iii. min_samples_split=
        iv. min_samples_leaf
    d. **Support Vector Machines**
        i. C
        ii. Epsilon
        iii. Gamma
    e. **Naïve Bayes**
        i. We have used Defaults values for
        ii. Priors
        iii. Var_smoothing
    f. **Neural Network**
        i. Random state = 42
        ii. Hidden layer sizes = 100,50
        iii. Max_iter = 1000

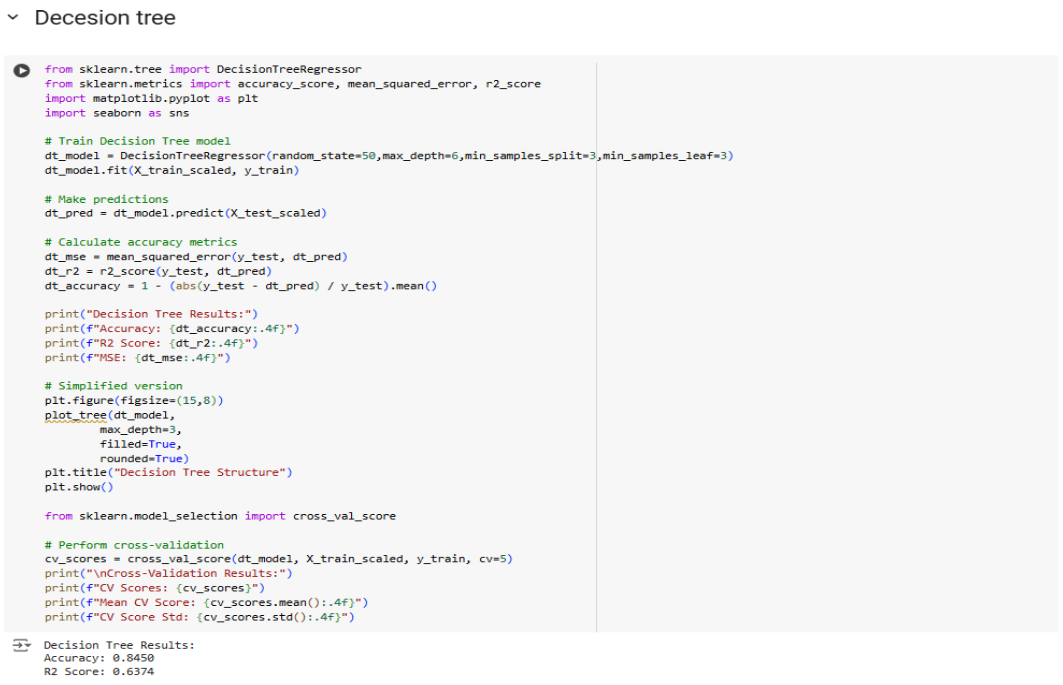
Hardware Description

We Performed the data modeling on hardware as below

Hardware Specifications:

- **Processor**: Intel Core i7
- **RAM**: 64GB
- **Storage**: SSD 1TB
- **Operating System**: Windows 11

5. **Discussing outcomes of data mining techniques.**
   a. **Decision Tree**
      i. In decision tree we have a flow chart like structure. We have each node as test and every branch as outcome of test and every leaf will be the predicted value.

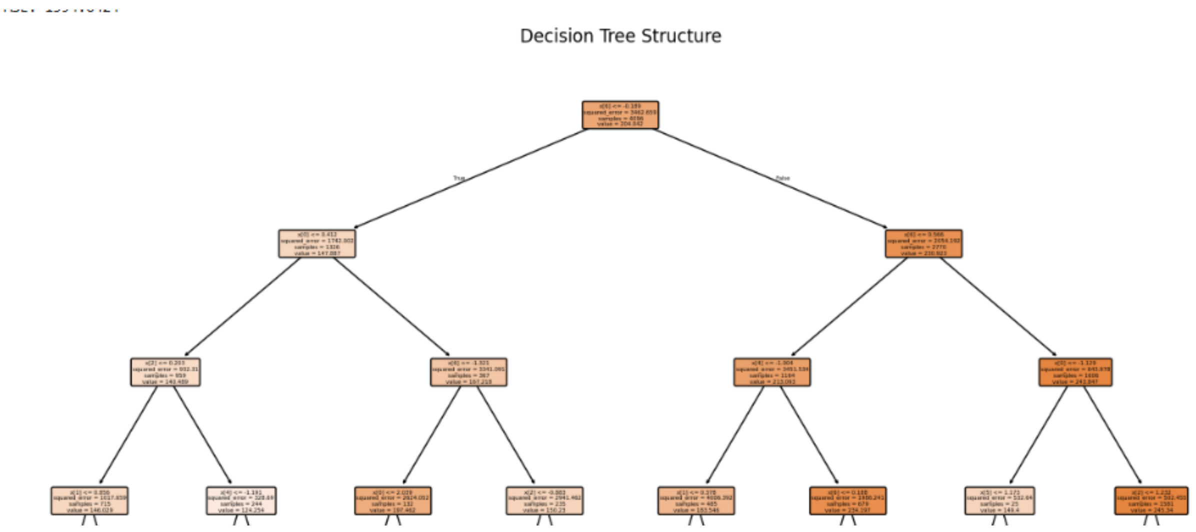


Decesion tree

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score, mean_squared_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns

# Train Decision Tree model
dt_model = DecisionTreeRegressor(random_state=50,max_depth=6,min_samples_split=3,min_samples_leaf=3)
dt_model.fit(X_train_scaled, y_train)

# Make predictions
dt_pred = dt_model.predict(X_test_scaled)

# Calculate accuracy metrics
dt_mse = mean_squared_error(y_test, dt_pred)
dt_r2 = r2_score(y_test, dt_pred)
dt_accuracy = 1 - (abs(y_test - dt_pred) / y_test).mean()

print("Decision Tree Results:")
print(f"Accuracy: {dt_accuracy:.4f}")
print(f"R2 Score: {dt_r2:.4f}")
print(f"MSE: {dt_mse:.4f}")

# Simplified version
plt.figure(figsize=(15,8))
plot_tree(dt_model,
        max_depth=3,
        filled=True,
        rounded=True)
plt.title("Decision Tree Structure")
plt.show()

from sklearn.model_selection import cross_val_score

# Perform cross-validation
cv_scores = cross_val_score(dt_model, X_train_scaled, y_train, cv=5)
print("\nCross-Validation Results:")
print(f"CV Scores: {cv_scores}")
print(f"Mean CV Score: {cv_scores.mean():.4f}")
print(f"CV Score Std: {cv_scores.std():.4f}")
```

```
Decision Tree Results:
Accuracy: 0.8450
R2 Score: 0.6374
```

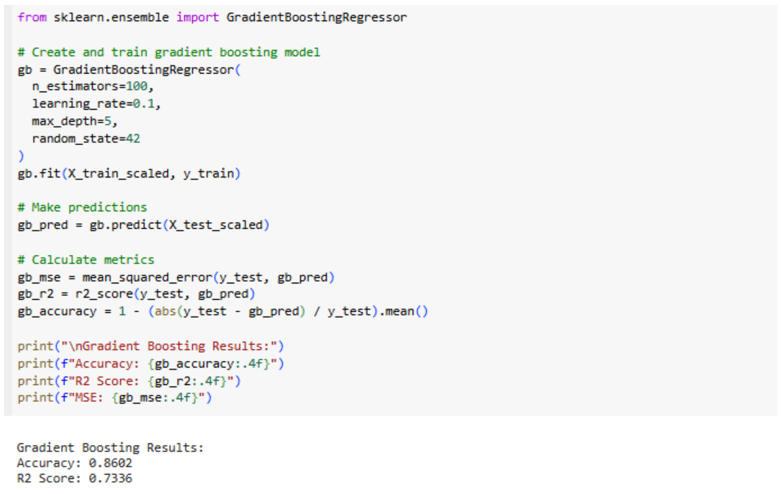
      ii. We got accuracy as 0.84 which indicates good prediction and got R square as 0.63 which is moderate MAPE as 14% indicating predictions are off by 14%.



b. Gradient Boosting

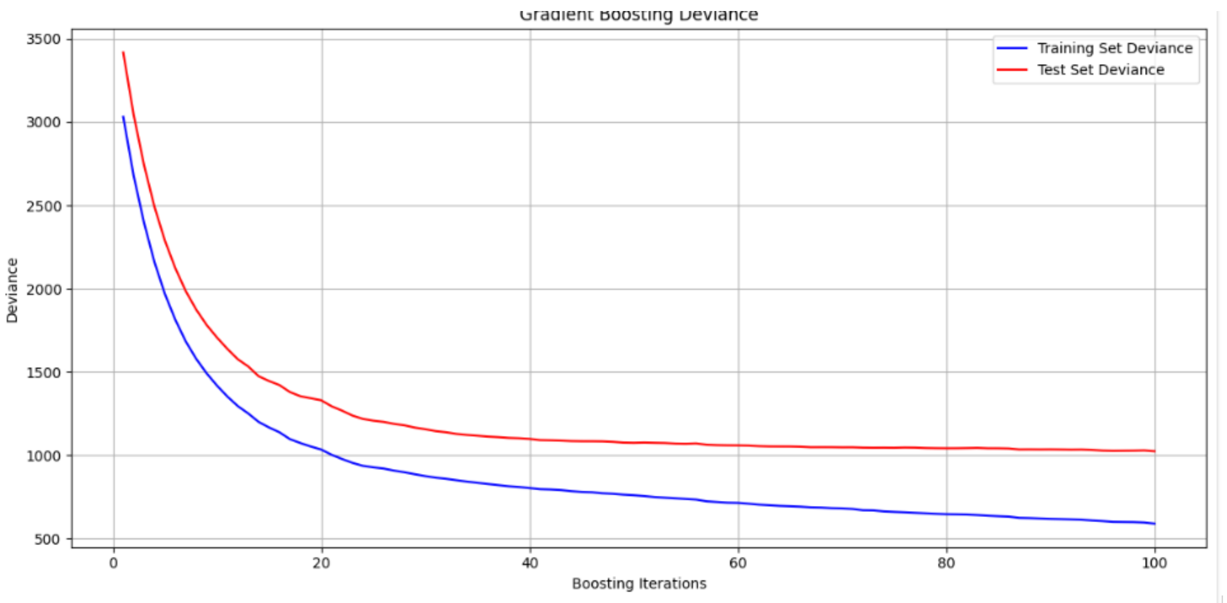
i. In this model will be built sequentially and every new model will correct their errors of previous model.

ii. In this model we got an accuracy of 0.86 which indicates it is predicting good and the R square value is 0.73 which is also good and MAPE value is also less which is 11% which also good indicating this model bitterly suits for the predction.



```python
from sklearn.ensemble import GradientBoostingRegressor

# Create and train gradient boosting model
gb = GradientBoostingRegressor(
  n_estimators=100,
  learning_rate=0.1,
  max_depth=5,
  random_state=42
)
gb.fit(X_train_scaled, y_train)

# Make predictions
gb_pred = gb.predict(X_test_scaled)

# Calculate metrics
gb_mse = mean_squared_error(y_test, gb_pred)
gb_r2 = r2_score(y_test, gb_pred)
gb_accuracy = 1 - (abs(y_test - gb_pred) / y_test).mean()

print("\nGradient Boosting Results:")
print(f"Accuracy: {gb_accuracy:.4f}")
print(f"R2 Score: {gb_r2:.4f}")
print(f"MSE: {gb_mse:.4f}")
```

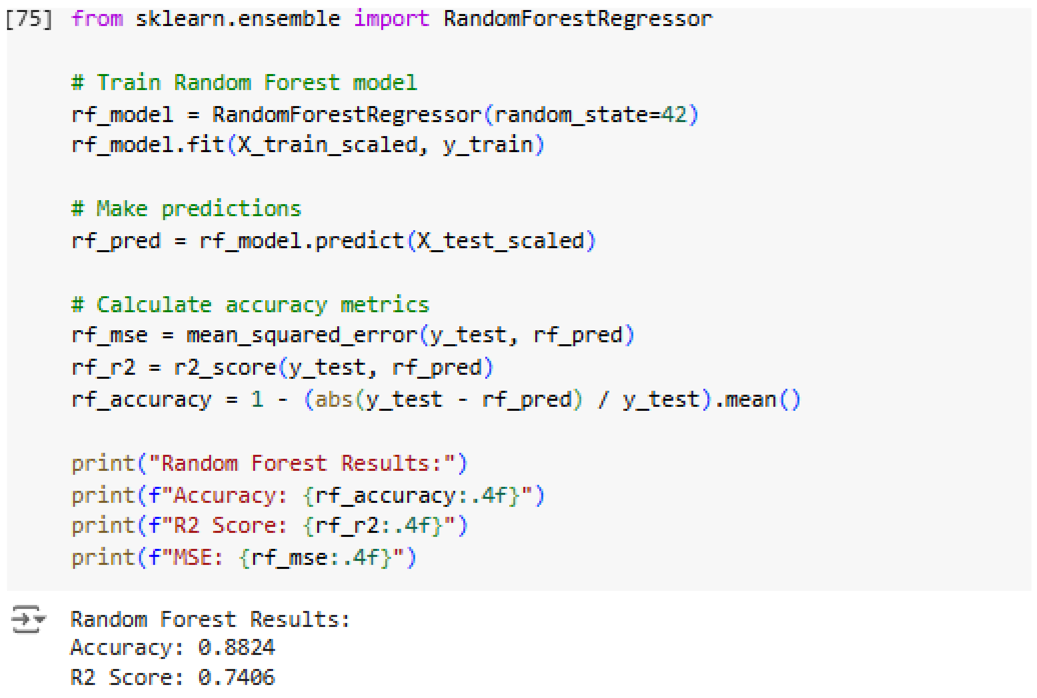
```
Gradient Boosting Results:
Accuracy: 0.8602
R2 Score: 0.7336
```



c. **Random Forest**

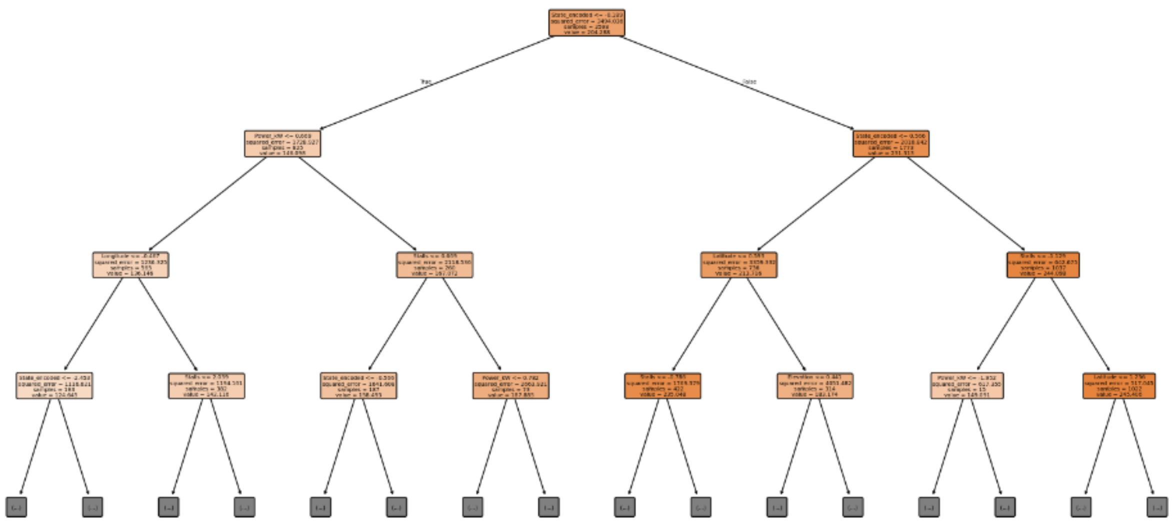
   i. In this model we will build multiple decision trees and select individual trees as output predicted ones.

   ii. In this model we got an accuracy of 0.88 which is higher than other models and is good and R square value is also more 0.74. and the MAPE is also less than other models indicating this model suits best in predicting the target variable.



```
[75] from sklearn.ensemble import RandomForestRegressor

     # Train Random Forest model
     rf_model = RandomForestRegressor(random_state=42)
     rf_model.fit(X_train_scaled, y_train)

     # Make predictions
     rf_pred = rf_model.predict(X_test_scaled)

     # Calculate accuracy metrics
     rf_mse = mean_squared_error(y_test, rf_pred)
     rf_r2 = r2_score(y_test, rf_pred)
     rf_accuracy = 1 - (abs(y_test - rf_pred) / y_test).mean()

     print("Random Forest Results:")
     print(f"Accuracy: {rf_accuracy:.4f}")
     print(f"R2 Score: {rf_r2:.4f}")
     print(f"MSE: {rf_mse:.4f}")
```

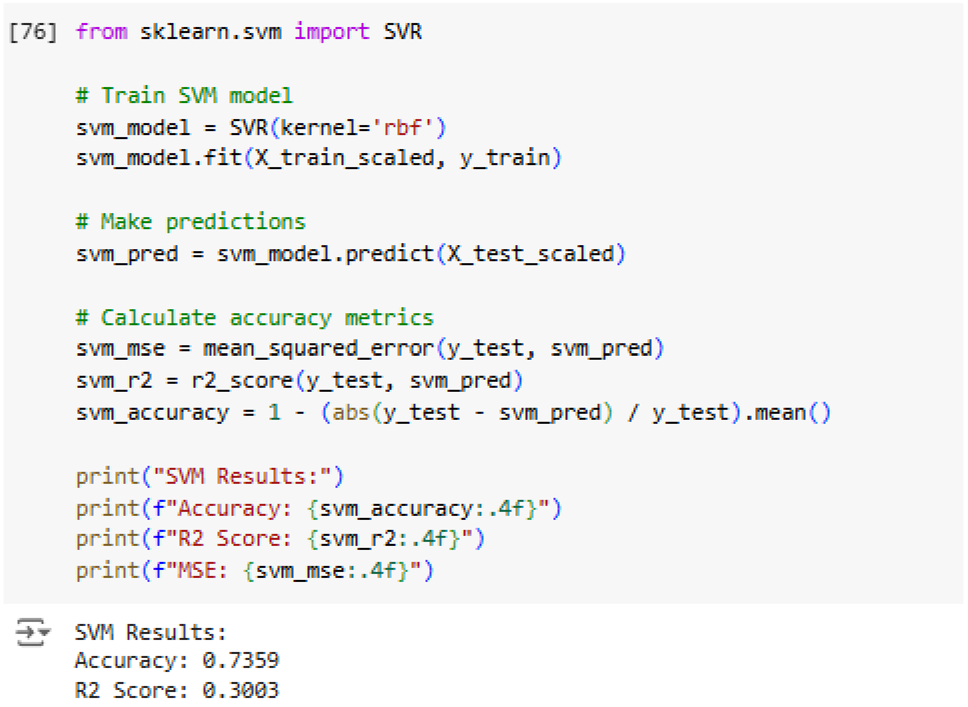
```
Random Forest Results:
Accuracy: 0.8824
R2 Score: 0.7406
```



d. SVM

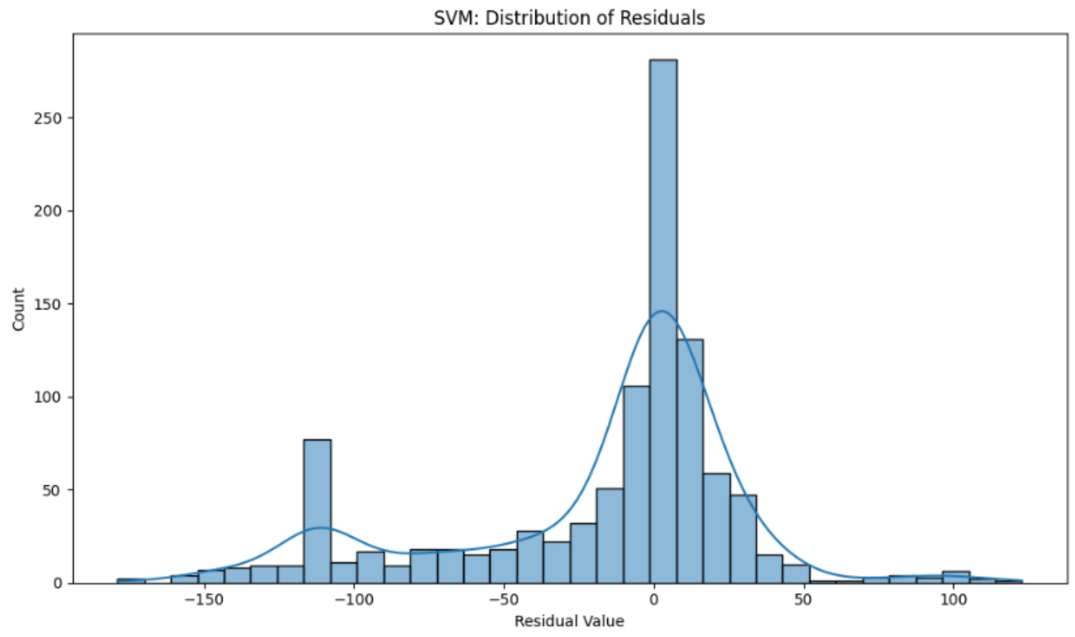
i. In this model we find the best hyperplane that separates the points in the high dimension space.

ii. In this model we got the accuracy of 0.73 and R square value of 0.30 which is very low suggesting poor performance of the model.



```
[76] from sklearn.svm import SVR

     # Train SVM model
     svm_model = SVR(kernel='rbf')
     svm_model.fit(X_train_scaled, y_train)

     # Make predictions
     svm_pred = svm_model.predict(X_test_scaled)

     # Calculate accuracy metrics
     svm_mse = mean_squared_error(y_test, svm_pred)
     svm_r2 = r2_score(y_test, svm_pred)
     svm_accuracy = 1 - (abs(y_test - svm_pred) / y_test).mean()

     print("SVM Results:")
     print(f"Accuracy: {svm_accuracy:.4f}")
     print(f"R2 Score: {svm_r2:.4f}")
     print(f"MSE: {svm_mse:.4f}")
```

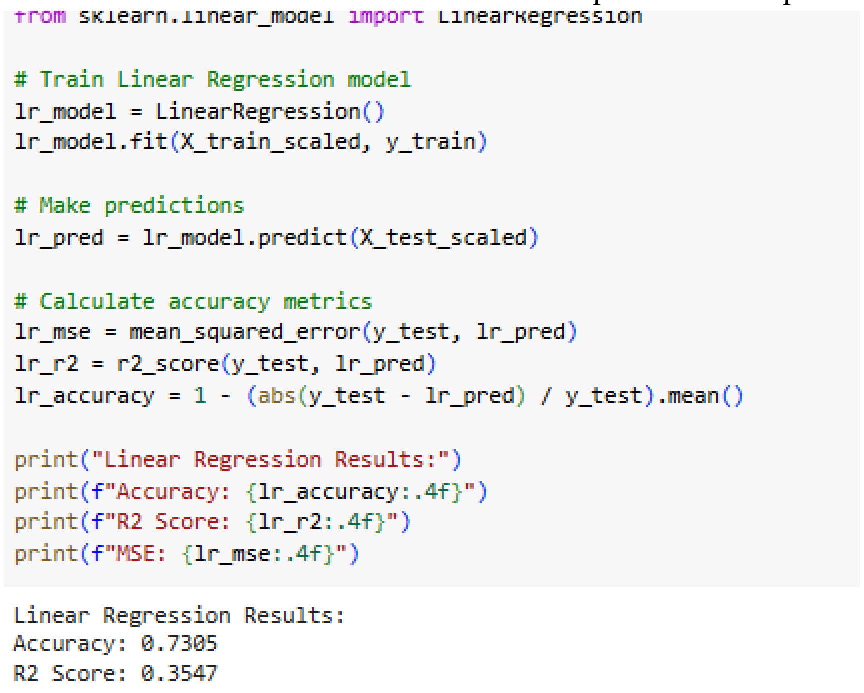
```
SVM Results:
Accuracy: 0.7359
R2 Score: 0.3003
```



e. **Linear Regression**

   i. In this model we build a statistical model for dependent and 1 or more independent values.

   ii. In this model we got the accuracy of 0.73 nd R square of 0.35 indicating this model is not best suited to predect the output.



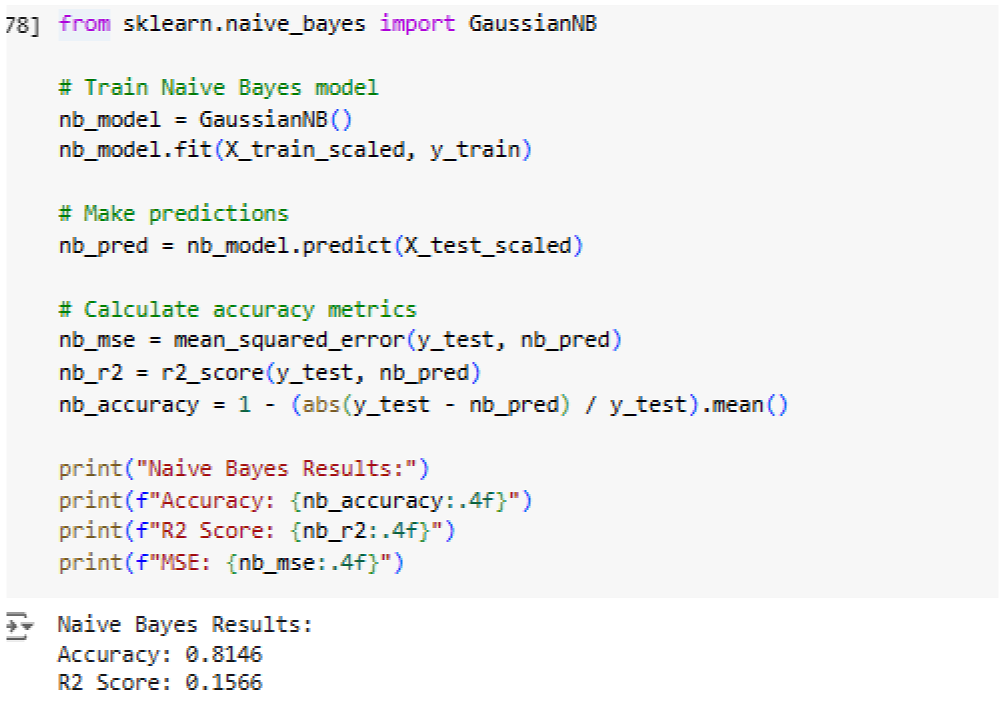
```
from sklearn.linear_model import LinearRegression

# Train Linear Regression model
lr_model = LinearRegression()
lr_model.fit(X_train_scaled, y_train)

# Make predictions
lr_pred = lr_model.predict(X_test_scaled)

# Calculate accuracy metrics
lr_mse = mean_squared_error(y_test, lr_pred)
lr_r2 = r2_score(y_test, lr_pred)
lr_accuracy = 1 - (abs(y_test - lr_pred) / y_test).mean()

print("Linear Regression Results:")
print(f"Accuracy: {lr_accuracy:.4f}")
print(f"R2 Score: {lr_r2:.4f}")
print(f"MSE: {lr_mse:.4f}")
```

```
Linear Regression Results:
Accuracy: 0.7305
R2 Score: 0.3547
```

f. Naïve Bayes

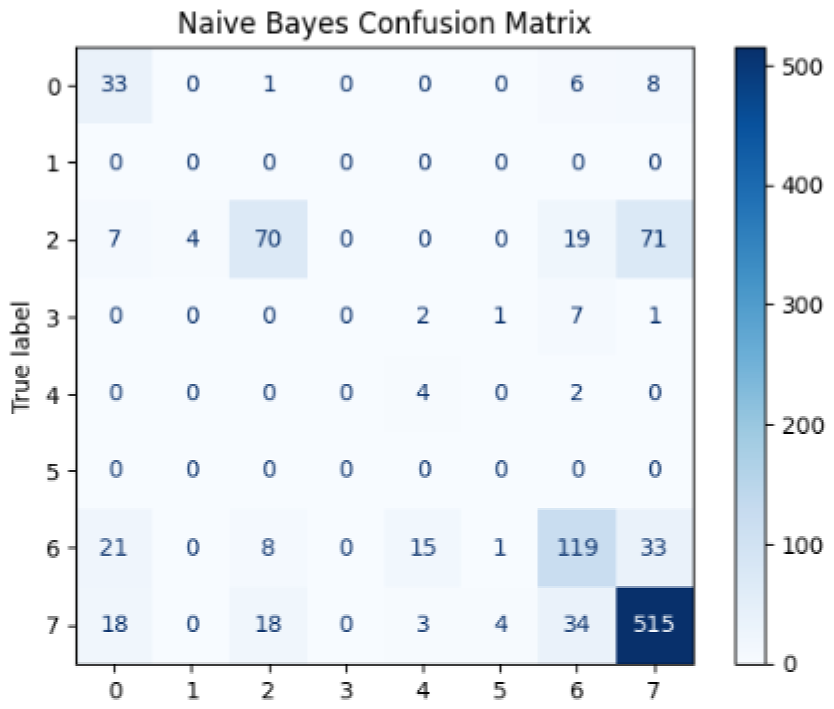
i. In this model we build probabilities based algorithms that are based on bayes theorem.

ii. In this model we got the accuracy of 0.81 and R square as very low indicating poor capability.



```
78] from sklearn.naive_bayes import GaussianNB

    # Train Naive Bayes model
    nb_model = GaussianNB()
    nb_model.fit(X_train_scaled, y_train)

    # Make predictions
    nb_pred = nb_model.predict(X_test_scaled)

    # Calculate accuracy metrics
    nb_mse = mean_squared_error(y_test, nb_pred)
    nb_r2 = r2_score(y_test, nb_pred)
    nb_accuracy = 1 - (abs(y_test - nb_pred) / y_test).mean()

    print("Naive Bayes Results:")
    print(f"Accuracy: {nb_accuracy:.4f}")
    print(f"R2 Score: {nb_r2:.4f}")
    print(f"MSE: {nb_mse:.4f}")
```

```
Naive Bayes Results:
Accuracy: 0.8146
R2 Score: 0.1566
```



	0	1	2	3	4	5	6	7
0	33	0	1	0	0	0	6	8
1	0	0	0	0	0	0	0	0
2	7	4	70	0	0	0	19	71
3	0	0	0	0	2	1	7	1
4	0	0	0	0	4	0	2	0
5	0	0	0	0	0	0	0	0
6	21	0	8	0	15	1	119	33
7	18	0	18	0	3	4	34	515

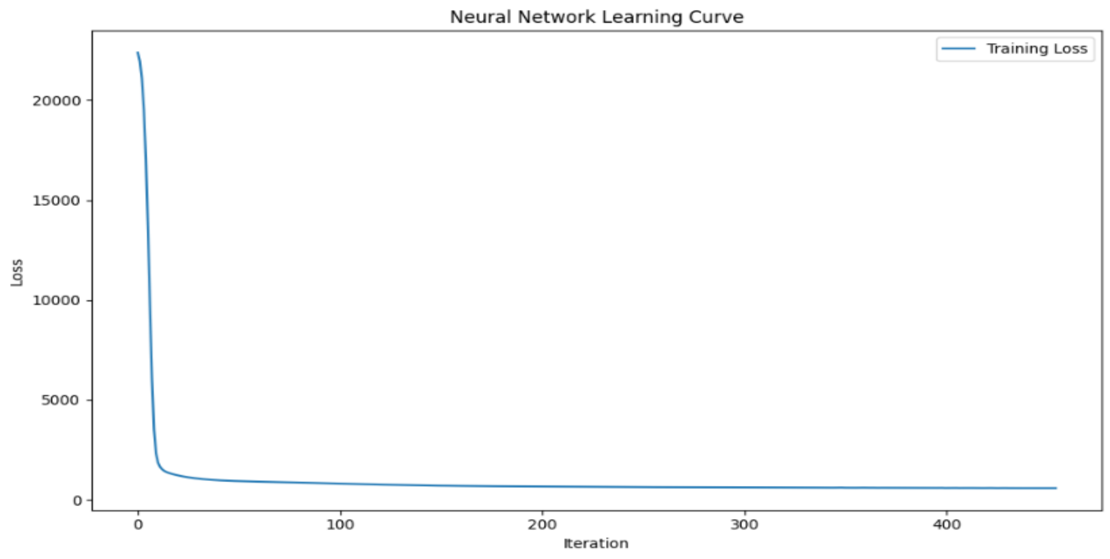
g. **Neural Network (MLP Regressor)**

   i. In this model we have multiple nodes/neurons. Where we use backpropagation for training and building complex relationships.

   ii. In this model we got the accuracy of 0.83 which is good and r square of 0.64 which is good and MAPE score of 12.50 which is less. So this model is good for predicting the output target variable.

```python
from sklearn.neural_network import MLPRegressor

# Train Neural Network model
nn_model = MLPRegressor(hidden_layer_sizes=(100, 50), max_iter=1000, random_state=42)
nn_model.fit(X_train_scaled, y_train)

# Make predictions
nn_pred = nn_model.predict(X_test_scaled)

# Calculate accuracy metrics
nn_mse = mean_squared_error(y_test, nn_pred)
nn_r2 = r2_score(y_test, nn_pred)
nn_accuracy = 1 - (abs(y_test - nn_pred) / y_test).mean()

print("Neural Network Results:")
print(f"Accuracy: {nn_accuracy:.4f}")
print(f"R2 Score: {nn_r2:.4f}")
print(f"MSE: {nn_mse:.4f}")
```

```
Neural Network Results:
Accuracy: 0.8366
R2 Score: 0.6408
```



Visualization Techniques:

1.Decision Tree visualization:

We have used it to show the hierarchical structure of decision tree.

2. Gradient Boosting learning curve:

We have used it to visualize convergence over overfitting and used dual line plot to check and compare the training vs testing.

3.Random Forest Tree Visualization:

We have visualized single tree from the random forest similar to the decision tree with feature names.

4.Confusion matrix:

We have used confusion matrix for naïve bayes model to show the true positive,false positive, true negatve , false negative.

5.Neural Network learning curve

We have used the learning curve for neural network to show the loss changes while training and used to monitor the learning process.

6. Conclusion

In the project we have performed different data mining techniques to answer our question i.e To find the power output demand of tesla supercharger locations based on different features. We have used Decision Trees, Gradient bossing, SVM, random forest, linear regression, naïve bayes, neural network. We calculates accuracy, R square score, MSE, etc...,

After analyzing everything we found that Random Forest is the best performing model with the highest accuracy of 0.88 and R square score of 0.74. Then we have the second best as gradient boosting with accuracy of 0.86 and R square score of 0.73. Then the Decision tree has good accuracy but does not perform well as it has less R square value. Similarly other models, if they have good accuracy, they have less R square value and vice versa.

To conclude The Models have successfully able to predect the power demand across different locations. And we found random forest was the most effective model which has provided good balance between accuracy and complexity. Further improvement could be hyperparameter tuning and optimizing the model more efficiently.