Phase 6 Optimization

Link to our github repository: https://github.com/hemanthreddy-1711/Data-Mining-Project-Unstoppable/tree/main

Google colab link for our work:
https://colab.research.google.com/drive/1pCakX6qblH0FlLqDPhAA6G8FWvlPA_Yn?usp=sharing

1. **Name of the team:** Unstoppable

   Team Member Names and emails:

   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**

   a. Random Forest
   b. Gradient Boosting Machines
   c. Support Vector Machines (SVM)

4. **Parameters and hyperparameters**

    a. Random Forest
        i. n_estimators – number of trees
        ii. max_depth – maximum depth of the tree
        iii. min_sample_split – minimum number of samples in internal node
        iv. min_samples_leaf – minimum number of samples in leaf

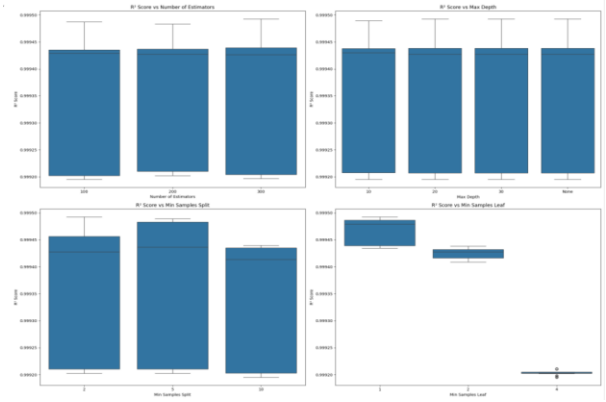
    b. Gradient Boosting
        i. n_estimators – number of stages
        ii. learning_rate – each tree contribution
        iii. max_depth – max depth of tree
        iv. min_samples_split – minimum number of samples in internal node

    c. Support Vector Machine
        i. C – trade of between training error and model complexity
        ii. Gamma – to influence the ability to generalize the training data

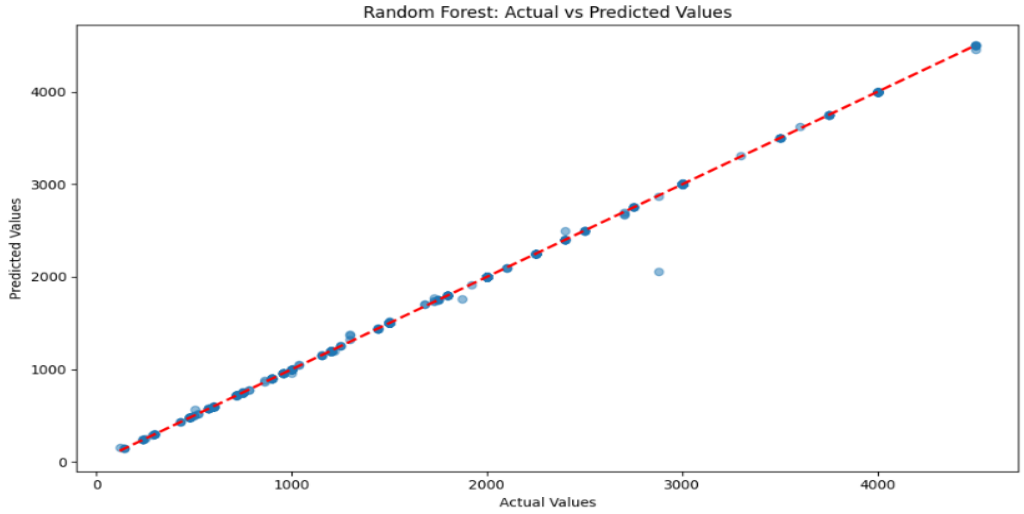
5. **List Techniques used to describe how your optimization techniques enhanced your data mining techniques outcomes from different perspectives and varied performance metrics**
    a. **Grid Search Cross Validation** (GridSearchCV): We have used GridSearch for specific parameter values and used cross validation to evaluate the performance for each combination.
    b. **randomizedSearchCV** method to evaluate specific parameters combination that we have used for the models which are used for testing and training of data. It will also help prevent overfitting.
    c. We are getting optimized parameters based on R2 score.
    d. **For Random Forest we have selected the parameters as:**
    n_estimators: [100, 200, 300]
    max_depth: [10, 20, 30, None]
    min_samples_split: [2, 5, 10]
    samples_leaf: [1, 2, 4]

```python
# 1. Random Forest Optimization
rf_param_grid = {
  'n_estimators': [100, 200, 300],
  'max_depth': [10, 20, 30, None],
  'min_samples_split': [2, 5, 10],
  'min_samples_leaf': [1, 2, 4]
}

rf_grid = GridSearchCV(RandomForestRegressor(random_state=42),
                       rf_param_grid, cv=5, scoring='r2')
rf_grid.fit(X_train, y_train)
rf_optimized = rf_grid.best_estimator_
rf_optimized_pred = rf_optimized.predict(X_test)
```



```
Best Parameters:
{'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300}
```

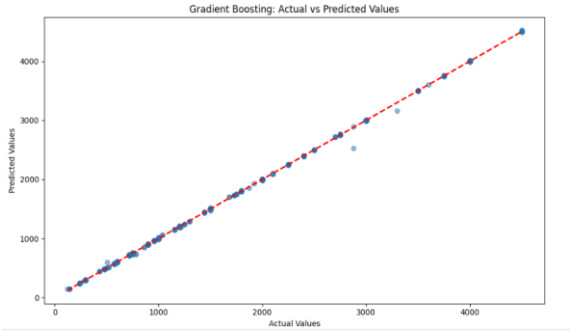


e. **For Gradient Boosting we have selected below parameters:**
   n_estimators: [100, 200, 300]
   learning_rate: [0.01, 0.1, 0.3]
   max_depth: [3, 4, 5]
   min_samples_split: [2, 4]

```
Best Parameters:
{'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300}
```

```python
# 2. Gradient Boosting Optimization
gb_param_grid = {
  'n_estimators': [100, 200, 300],
  'learning_rate': [0.01, 0.1, 0.3],
  'max_depth': [3, 4, 5],
  'min_samples_split': [2, 4]
}

gb_grid = GridSearchCV(GradientBoostingRegressor(random_state=42),
                       gb_param_grid, cv=5, scoring='r2')
gb_grid.fit(X_train, y_train)
gb_optimized = gb_grid.best_estimator_
gb_optimized_pred = gb_optimized.predict(X_test)
```

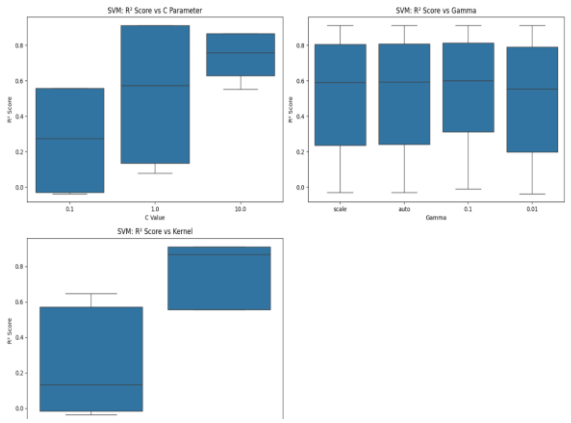


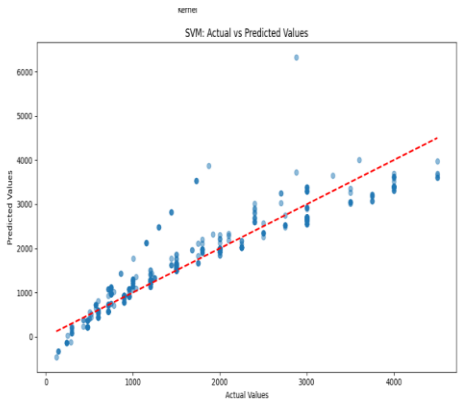
f. **For SVM we have selected below parameters**:
C: [0.1, 1, 10]
gamma: ['scale', 'auto', 0.1, 0.01]
kernel: ['rbf', 'linear']

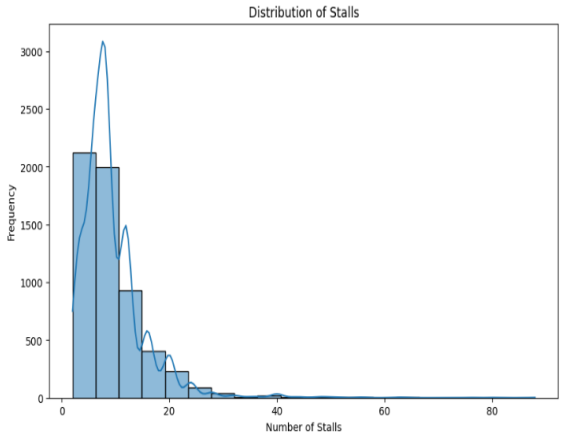
```
SVM Best Parameters:
{'C': 1, 'gamma': 'scale', 'kernel': 'linear'}
```

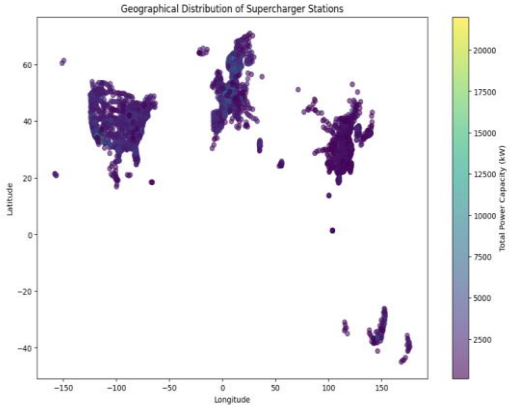
```
# 3. SVM Optimization
svm_param_grid = {
  'C': [0.1, 1, 10],
  'gamma': ['scale', 'auto', 0.1, 0.01],
  'kernel': ['rbf', 'linear']
}

svm_grid = GridSearchCV(SVR(), svm_param_grid, cv=5, scoring='r2')
svm_grid.fit(X_train, y_train)
svm_optimized = svm_grid.best_estimator_
svm_optimized_pred = svm_optimized.predict(X_test)
```









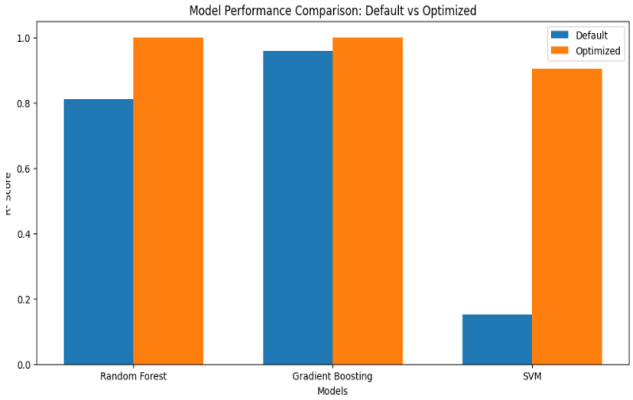
6. **Describe How The optimization technique has improved our models:**

R^2 Score Improvements:

The Random Forest has improved it's R2 score from 0.811 to 0.99 and Gradient Boosting has improved from 0.95 to 0.99 and SVM has improve from 0.15 to 0.90

Similarly The Random Forest has reduced it's MSE score from 682 to 657 and Gradient Boosting MSE Reduced from 349 to 156 and SVM has also reduced.

Similarly The RMSE and MAE values also have reduced indicating in better performance after the optimization.



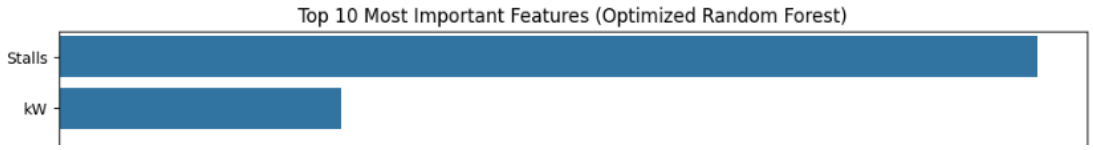

```
# Calculate optimized metrics
optimized_metrics = {
  'Random Forest': calculate_metrics(y_test, rf_optimized_pred),
  'Gradient Boosting': calculate_metrics(y_test, gb_optimized_pred),
  'SVM': calculate_metrics(y_test, svm_optimized_pred)
}

# Display optimized metrics
optimized_df = pd.DataFrame(optimized_metrics).round(4)
print("\nOptimized Model Metrics:")
print(optimized_df)
```

```
Optimized Model Metrics:
      Random Forest  Gradient Boosting        SVM
R2           0.9993             0.9998     0.9027
MSE         657.8305           156.7533  93993.9994
RMSE         25.6482            12.5201    306.5844
MAE           1.6297             2.3004    189.6670
```

```
# Feature importance analysis with optimized Random Forest
feature_importance = pd.DataFrame({
  'feature': X.columns,
  'importance': rf_optimized.feature_importances_
})
feature_importance = feature_importance.sort_values('importance', ascending=False)

plt.figure(figsize=(12, 6))
sns.barplot(data=feature_importance.head(10), x='importance', y='feature')
plt.title('Top 10 Most Important Features (Optimized Random Forest)')
plt.show()
```



Conclusion:

By using the GridSearchCV we have improved the performance of our model Significantly. We have identified the optimal hyperparameters for the models. The models Random forest has improved R^2 scores from 0.81 to 0.99 which indicated the increase in the ability for explaining the variance of target variable. In the same it has improved the scores of gradient boosting model and SVM model from 0.95 to 0.99 and from 0.15 to 0.90 respectively. We also identified the importance of the top 10 features in them stalls and KW are very important. And we have compared the models performances.

So Overall The GridSearchCV has allowed us to test combinations of hyperparameters, from which we got accurate models that has helped us to give more reliable prediction's for our research question.