Random Forest

The Sentinels

Tristen Brewer

Nicholas Fenech

Janeel Abrahams

What is a Random Forest?

- An ensemble technique
    - Multiple decision trees
        - Bagging :Bootstrap and Aggregation
    - Classification and **Regression**

Advantages vs Disadvantages

Advantages

It is easy to use and less sensitive to the training data compared to the decision tree.

It is more accurate than the decision tree algorithm.

It is effective in handling large datasets that have many attributes.

It can handle missing data, outliers, and noisy features.

Disadvantages

The model can be difficult to interpret.

May require expert-level insights when selecting parameter.

It's computationally, expensive.

Possible overfitting for complex models or high number of trees

What are the steps involved in building a Random Forest model?

Create

```
model
= RandomForestRegressor(
n_estimators = 100,
random_state =42)
```

Fit

Random Forest Jargons

- Bagging
- Out-Of-Bag error (OOB error)
- Gini Index

How does it differ from other machine learning algorithms?

- Ensemble Method
- Feature Selection
- Bootstrap Aggregating
- Non-parametric Model

What does the Target Data Set look like?

- Cars93 (Provided for the Assessment)
- Housing Data Set(Used as our Toy Data Set)
- 100,000 UK Used Cars (Real-world Data Set)

- Continuous Targets were used through all sets above:
- Price
- MPG

- Did not use any Categorical Variables

How is the model evaluated?

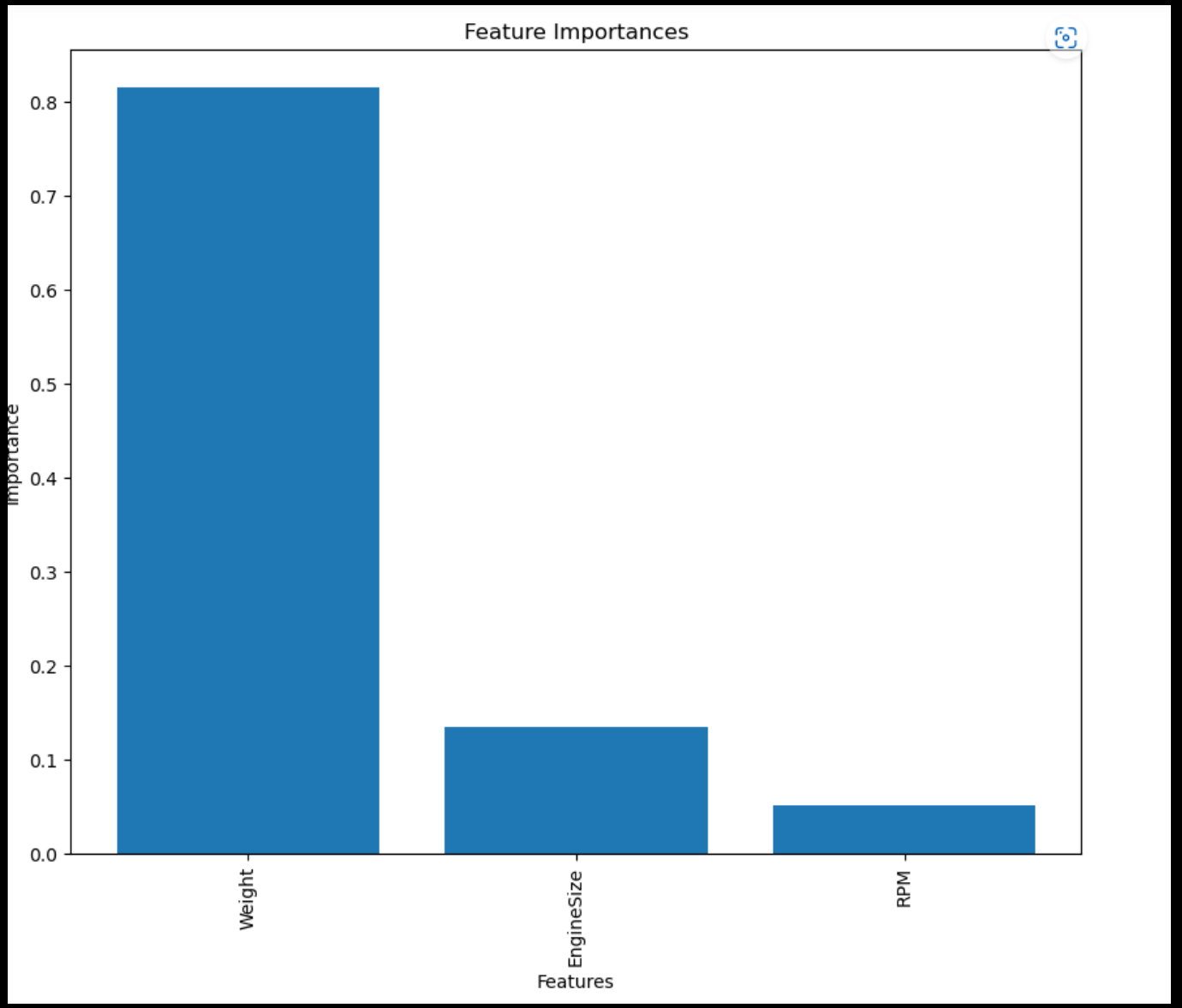
- R Squared Score
- Mean Absolute Error
- Mean Squared Error

Random Forest Hyperparameters

- Boot Strapping
- Feature Selection
- Depth
- Number of Trees
- and many others!

Random Forest at a Basic Level – Cars93

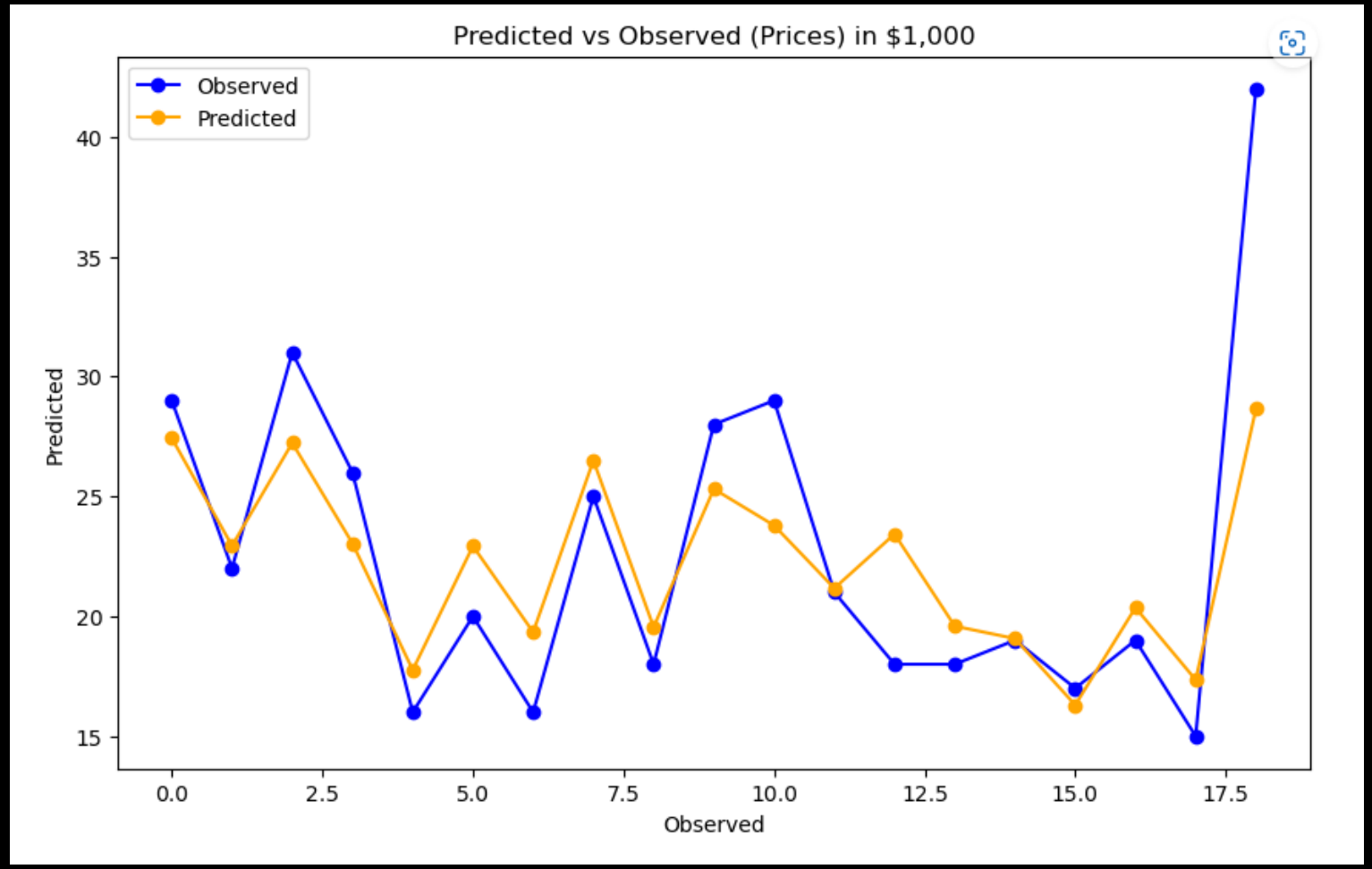
Feature importance



Random Forest at a Basic Level – Cars93

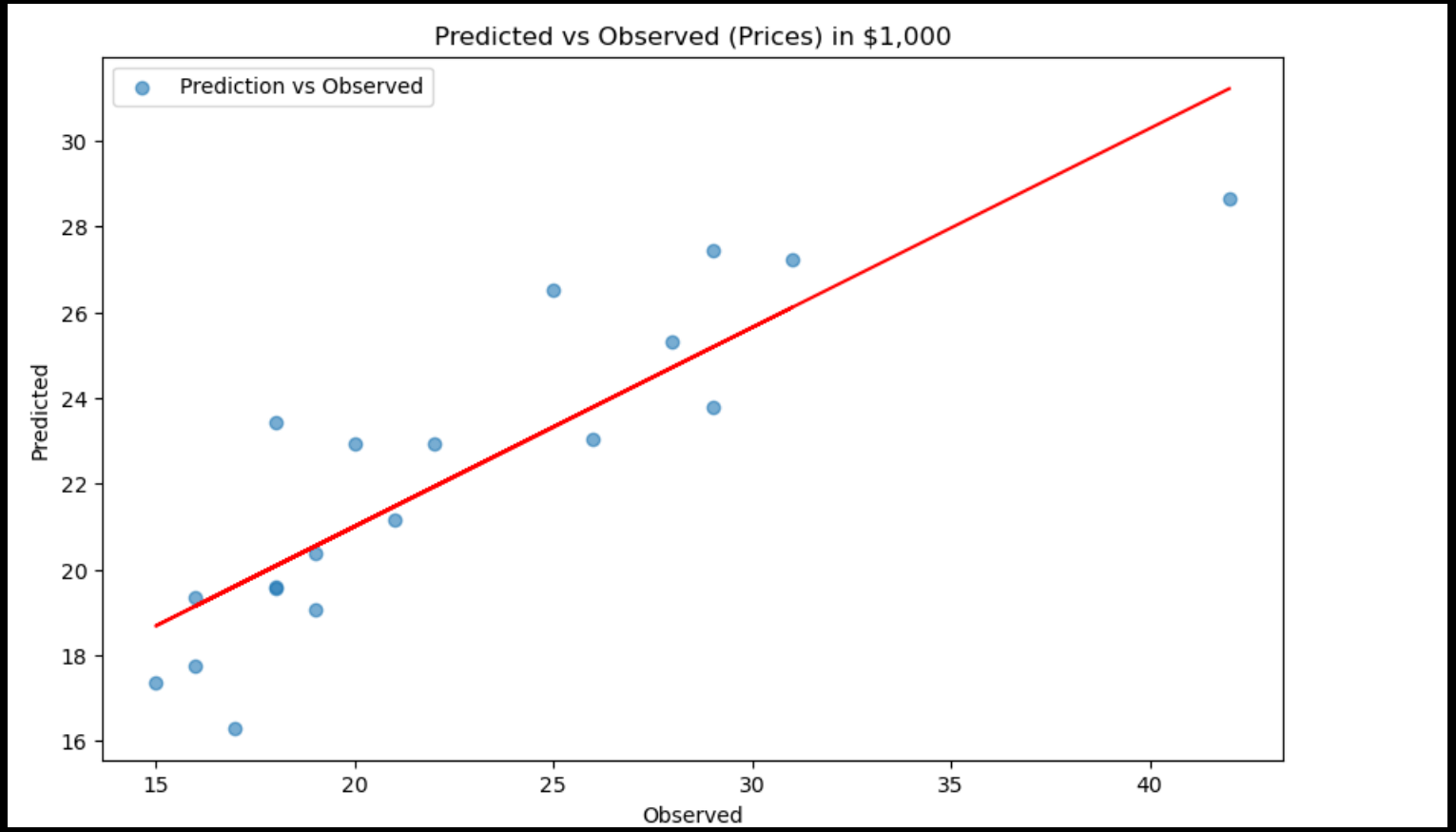
Line Graph

R2_score
Training- 0.97
Testing-0.64



Random Forest at a Basic Level – Cars93

Scatterplot



R2_score
Training- 0.97
Testing-0.64

Adding Hyperparameters – Housing Data

```
model = RandomForestRegressor( random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
train_score = model.score(X_train, y_train)
test_score = model.score(X_test, y_test)
mae = mean_absolute_error(y_test, y_pred)
print(f'R score train: {train_score}, R score test: {test_score} and Mae: {mae}')
```

✓ 0.7s

```
R score train: 0.9633544308793284, R score test: 0.7030596645042124 and Mae: 26535.530444155844
```

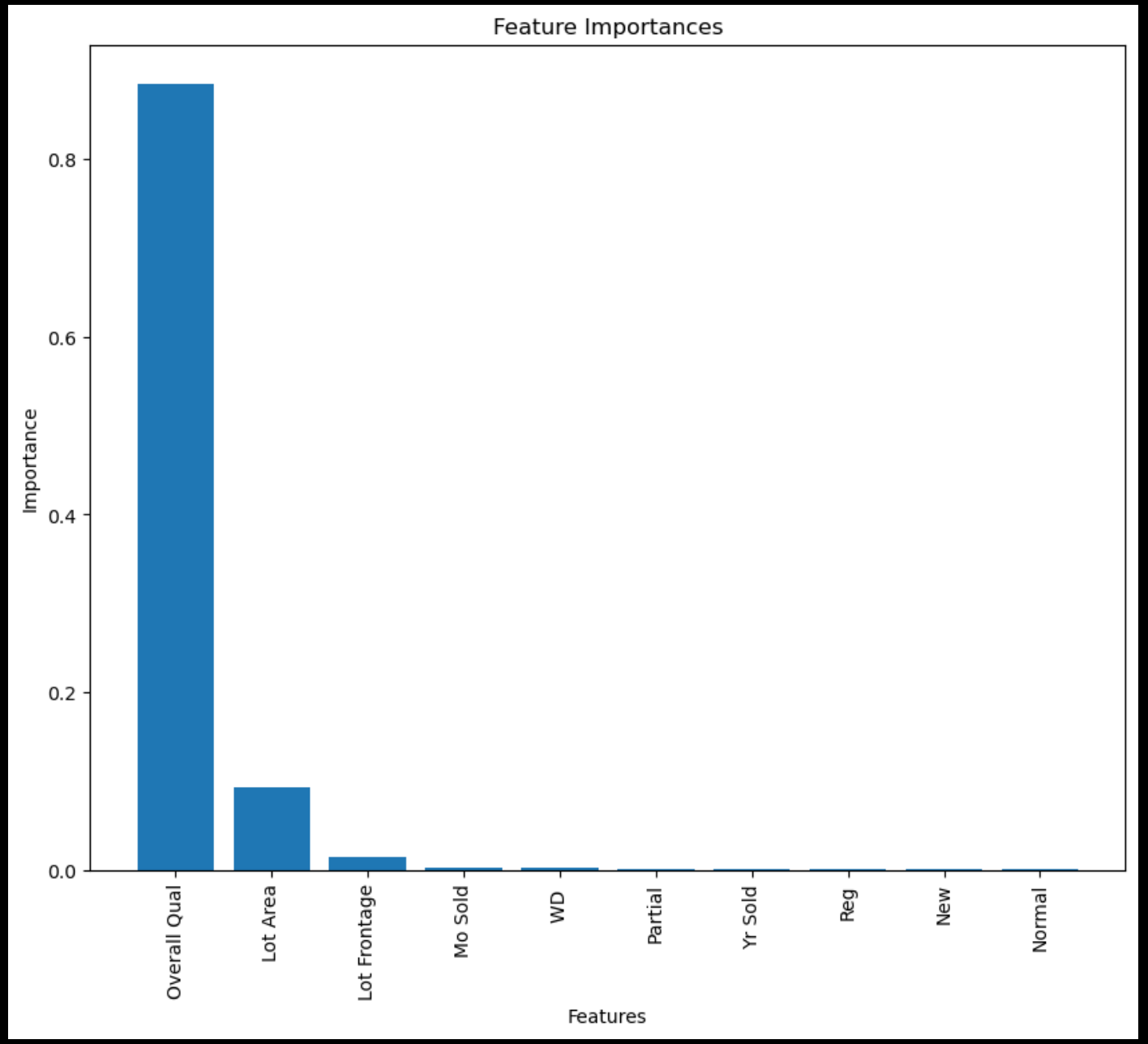
```
model = RandomForestRegressor(n_estimators = 300, max_depth = 4, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
train_score = model.score(X_train, y_train)
test_score = model.score(X_test, y_test)
mae = mean_absolute_error(y_test, y_pred)
print(f'R score train: {train_score}, R score test: {test_score} and Mae: {mae}')
```

✓ 0.8s

```
R score train: 0.8002909000927194, R score test: 0.7165861472271403 and Mae: 27199.65736347619
```

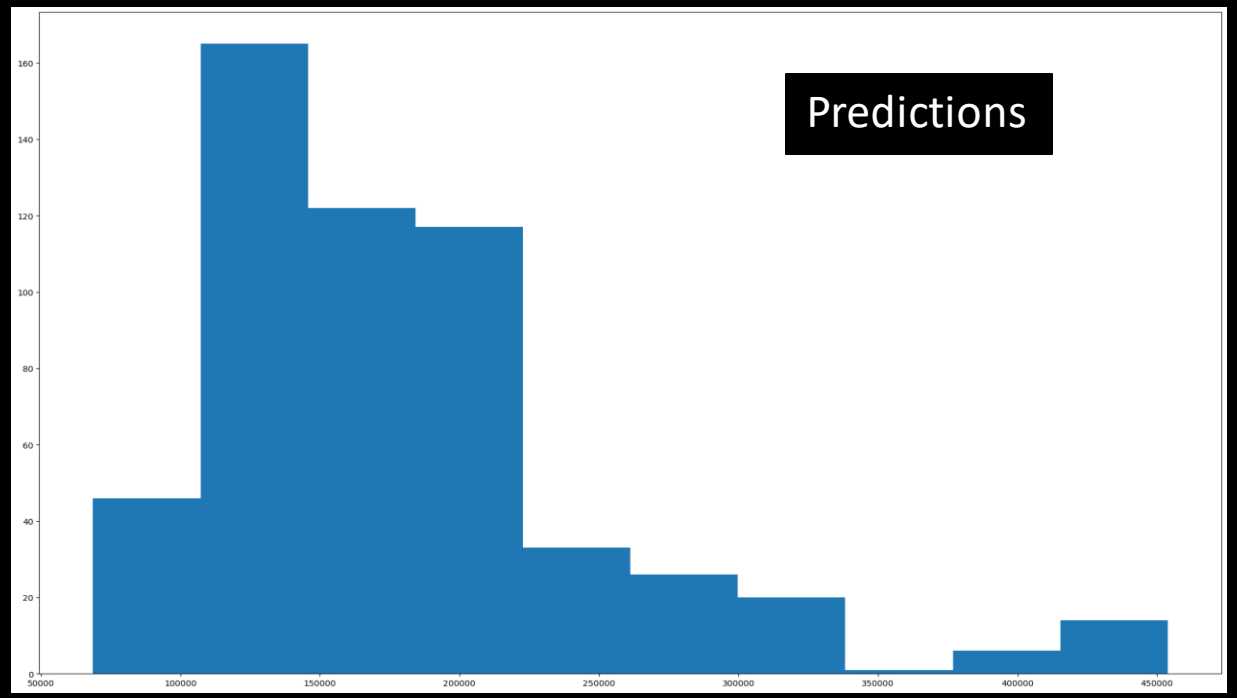
Random Forest Hyperparameters – Housing Data

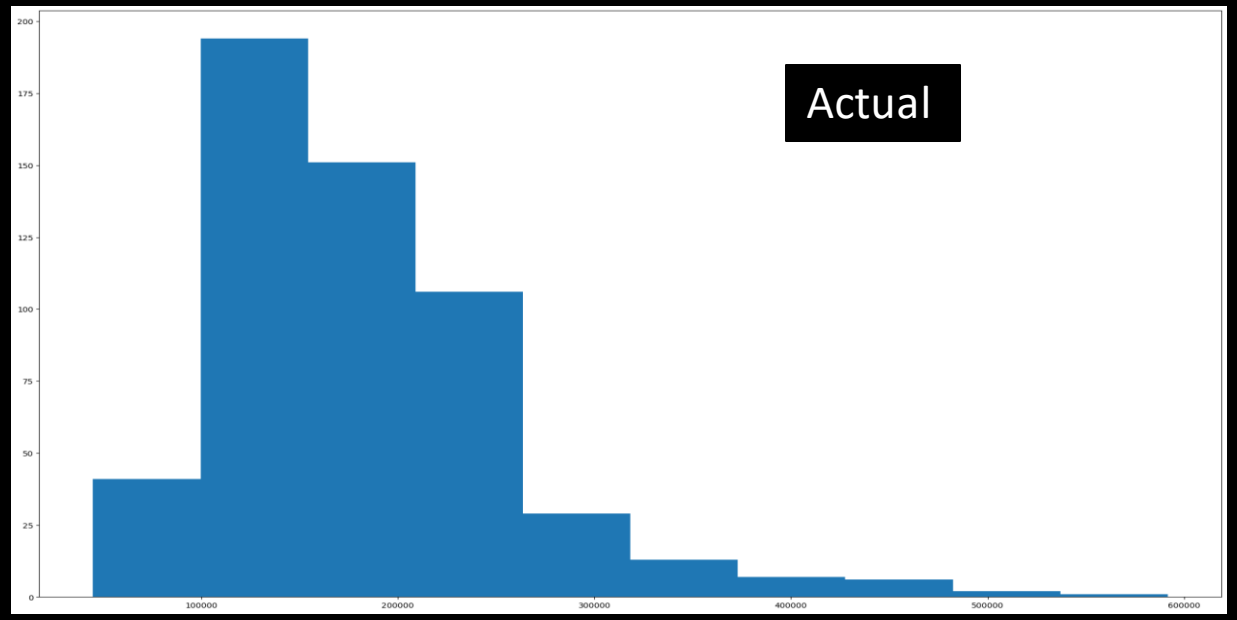
Feature Importance



Random Forest Hyperparameters – Housing Data

Histograms





Tuning Hyperparameters for Maximum Potential – Real World Cars

```python
from sklearn.model_selection import RandomizedSearchCV
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
# Method of selecting samples for training each tree
bootstrap = [True, False]
# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'bootstrap': bootstrap}
print(random_grid)
```

Python

```python
rf = RandomForestRegressor()

rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid, n_iter = 10, cv = 2, verbose=2, random_state=42, n_jobs = -1)

rf_random.fit(X_train, y_train)

rf_random.best_params_

# {'n_estimators': 800,
#  'max_features': 'sqrt',
#  'max_depth': 40,
#  'bootstrap': True}
```

Python

Tuning Hyperparameters for Maximum Potential – Real World Cars-Cont'd

```python
from sklearn.model_selection import GridSearchCV
# Create the parameter grid based on the results of random search
param_grid = {
    'bootstrap': [True],
    'max_depth': [30, 40, 60],
    'max_features': ['sqrt'],
    'n_estimators': [600, 800, 1000, 1400]
}
# Create a based model
rf = RandomForestRegressor()

# Instantiate the grid search model
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                           cv = 7, n_jobs = -1, verbose = 2)
```

Python

```python
grid_search.fit(X_train, y_train) # Fit the GridSearchCV on the training data

# Print the best parameters and best score found by GridSearchCV
print("Best parameters found: ", grid_search.best_params_)
print("Best score found: ", grid_search.best_score_)
```

Python

```
Fitting 7 folds for each of 12 candidates, totalling 84 fits
Best parameters found:  {'bootstrap': True, 'max_depth': 30, 'max_features': 'sqrt', 'n_estimators': 800}
Best score found:  0.9078030165191303
```

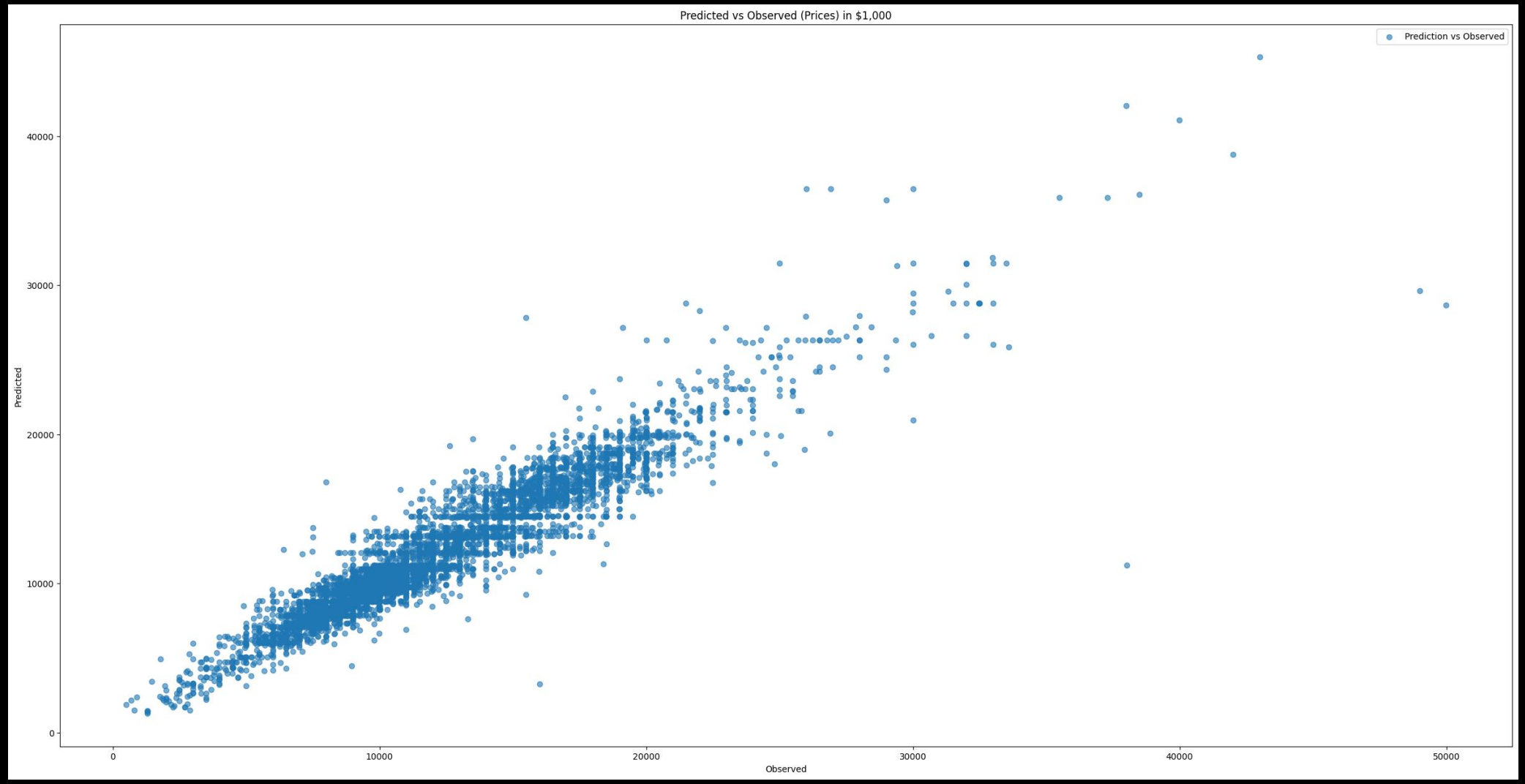
```python
final_model = RandomForestRegressor(bootstrap = True, max_depth = 30, max_features= 'sqrt', n_estimators= 800, oob_score= True)
final_model.fit(X_train, y_train)
```

20.1s

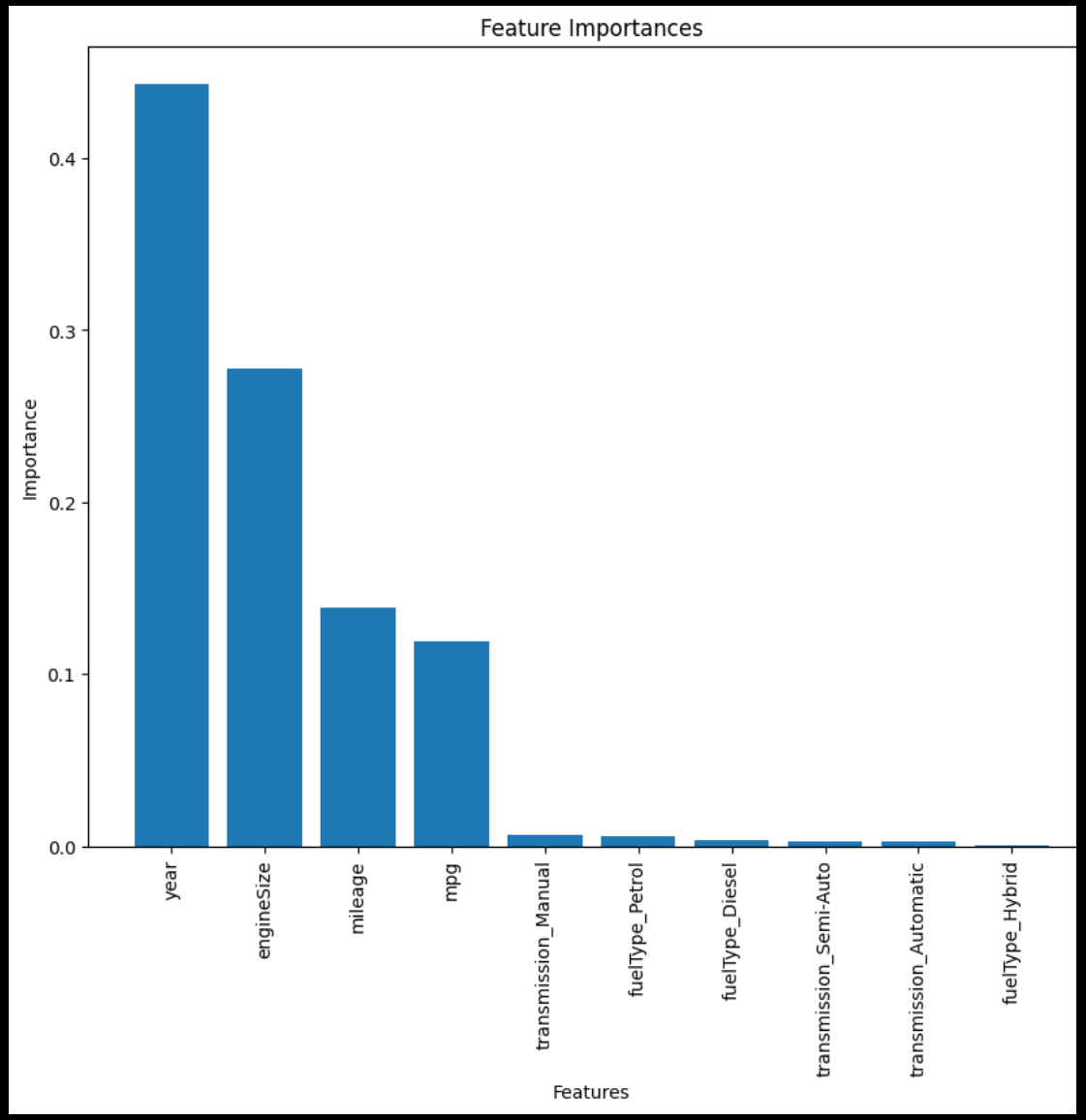
Python

Tuning Hyperparameters for Maximum Potential – Real World Cars- Cont'd

- Scatter Plot Trend
- Pre-tuned R-Squared = 89%
- Post-tuned R-Squared = 89%
- Mean Absolute Error = $991

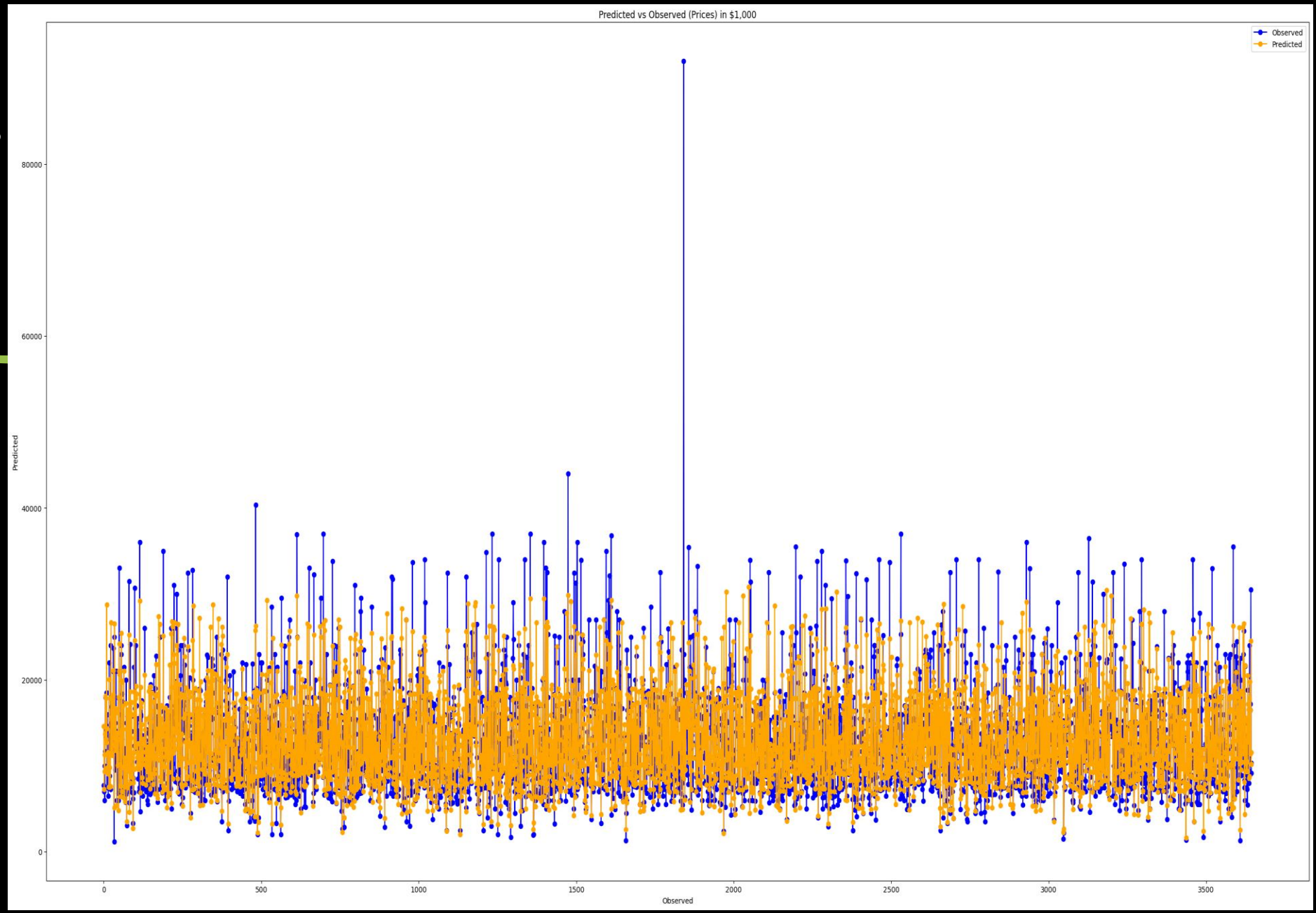


Tuning Hyperparameters for Maximum Potential – Real World Cars-Cont'd



Tuning Hyperparameters for Maximum Potential – Real World Cars-Cont'd

- Final Model used against an unseen dataset
- R-Squared = 65%
- Mean Absolute Error = 2441



What are some real-world applications of Random Forests?

- Churn prediction
- Medical diagnosis
- Fraud detection
- Environmental modeling
- Recommender systems

References

- Hyperparameter Tuning the Random Forest in Python: https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74
- Random Forest Regression with Python: https://www.youtube.com/watch?v=LhBOVWSu-tI
- Understand Random Forest Algorithms With Examples (Updated 2023): https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/
- Random forest for imbalanced datasets: https://towardsdatascience.com/random-forest-for-imbalanced-datasets-9d0c8dcb1eb7
- Preventing overfitting in random forest: https://towardsdatascience.com/preventing-overfitting-in-random-forest-8e7edffc2bbc
- Real-world applications of random forest: https://www.analyticsvidhya.com/blog/2017/09/30-questions-test-knowledge-random-forest-algorithm/
- Challenges of using random forest: https://towardsdatascience.com/3-challenges-of-random-forest-and-how-to-overcome-them-7dde3f50f26c
- Scikit Learn : https://scikit-learn.org/stable/modules/ensemble.html#random-forestshttps://scikit-learn.org/stable/modules/ensemble.html#random-forests

Moving to GitHub