Name : Keith Fernandes Roll No. : 25

St. Francis Institute of Technology, Mumbai-400 103
Department Of Information Technology

A.Y. 2025-2026
Class: BE-ITA/B, Semester: VII
Subject: Data Science Lab

Experiment – 8

1. **Aim:** To implement Supervised Learning algorithm - Random Forest.
2. **Objectives:** Students should be familiarize with Learning Architectures and Frameworks
3. **Prerequisite:** Python basics

4. **Pre-Experiment Exercise:**
   Theory:
   Random Forest Algorithm
   Decision trees involve the greedy selection of the best split point from the dataset at each step.

   This algorithm makes decision trees susceptible to high variance if they are not pruned. This high variance can be harnessed and reduced by creating multiple trees with different samples of the training dataset (different views of the problem) and combining their predictions. This approach is called bootstrap aggregation or bagging for short.

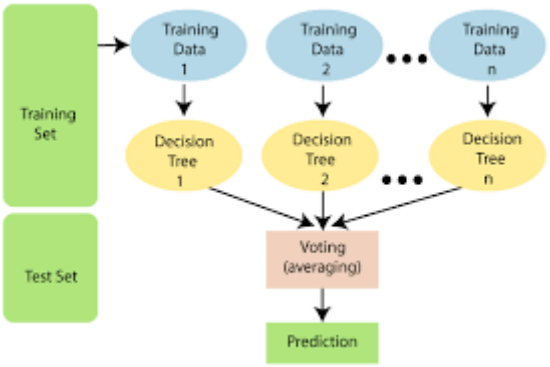
   A limitation of bagging is that the same greedy algorithm is used to create each tree, meaning that it is likely that the same or very similar split points will be chosen in each tree making the different trees very similar (trees will be correlated). This, in turn, makes their predictions similar, mitigating the variance originally sought.

   We can force the decision trees to be different by limiting the features (rows) that the greedy algorithm can evaluate at each split point when creating the tree. This is called the Random Forest algorithm.

   Like bagging, multiple samples of the training dataset are taken and a different tree trained on each. The difference is that at each point a split is made in the data and added to the tree, only a fixed subset of attributes can be considered.

   For classification problems, the type of problems we will look at in this tutorial, the number of attributes to be considered for the split is limited to the square root of the number of input features.

   num_features_for_split = sqrt(total_input_features)
   The result of this one small change are trees that are more different from each other (uncorrelated) resulting predictions that are more diverse and a combined prediction that often has better performance that single tree or bagging alone.



6. Laboratory Exercise

Procedure

i. Use google colab for programming.
ii. Import required packages.
iii. Demonstrate random forest classifier for any given dataset.
iv. Add relevant comments in your programs and execute the code. Test it for various cases.

Post-Experiments Exercise:

A. Extended Theory:

a. Write real life applications of Random Forest Classifier.

B. Conclusion:

1. Write what was performed in the program (s) .
2. What is the significance of program and what Objective is achieved?

C. References:

[1] https://machinelearningmastery.com/implement-random-forest-scratch-python/.
[2] https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/

Name: Keith Fernandes Roll No : 25

```
import pandas as pd

df = pd.read_csv('data.csv')
display(df.head())
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	street	city	statezip	country
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	1340	0	1955	2005	18810 Densmore Ave N	Shoreline	WA 98133	USA
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	3370	280	1921	0	709 W Blaine St	Seattle	WA 98119	USA
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	1930	0	1966	0	26206-26214 143rd Ave SE	Kent	WA 98042	USA
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	1000	1000	1963	0	857 170th Pl NE	Bellevue	WA 98008	USA
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	1140	800	1976	1992	9105 170th Ave NE	Redmond	WA 98052	USA

```
X = df.drop('price', axis=1)
y = df['price']
display(X.head())
display(y.head())
```

	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	street	city	statezip	country
0	2014-05-02 00:00:00	3.0	1.50	1340	7912	1.5	0	0	3	1340	0	1955	2005	18810 Densmore Ave N	Shoreline	WA 98133	USA
1	2014-05-02 00:00:00	5.0	2.50	3650	9050	2.0	0	4	5	3370	280	1921	0	709 W Blaine St	Seattle	WA 98119	USA
2	2014-05-02 00:00:00	3.0	2.00	1930	11947	1.0	0	0	4	1930	0	1966	0	26206-26214 143rd Ave SE	Kent	WA 98042	USA
3	2014-05-02 00:00:00	3.0	2.25	2000	8030	1.0	0	0	4	1000	1000	1963	0	857 170th Pl NE	Bellevue	WA 98008	USA
4	2014-05-02 00:00:00	4.0	2.50	1940	10500	1.0	0	0	4	1140	800	1976	1992	9105 170th Ave NE	Redmond	WA 98052	USA

	price
0	313000.0
1	2384000.0
2	342000.0
3	420000.0
4	550000.0

dtype: float64

Name: Keith Fernandes Roll No : 25

```
from sklearn.model_selection import train_test_split

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

display(X_train.head())
display(X_test.head())
display(y_train.head())
display(y_test.head())
```

	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	street	city	statezip	country
1898	2014-06-04 00:00:00	4.0	2.50	2770	45514	2.0	0	0	4	2770	0	1989	0	18630 NE 202nd St	Woodinville	WA 98077	USA
1370	2014-05-27 00:00:00	4.0	3.00	3720	29043	2.0	0	0	3	3720	0	1991	0	10161 134th Pl NE	Kirkland	WA 98033	USA
3038	2014-06-23 00:00:00	4.0	2.50	2810	11120	2.0	0	0	3	2810	0	1982	0	22120 NE 26th Pl	Sammamish	WA 98074	USA
2361	2014-06-12 00:00:00	4.0	3.75	4030	10800	2.0	0	0	3	4030	0	2006	0	619 9th Ave	Kirkland	WA 98033	USA
156	2014-05-06 00:00:00	3.0	2.00	2000	7000	2.0	0	0	3	2000	0	1916	1986	6422 Marshall Ave SW	Seattle	WA 98136	USA

	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	street	city	statezip	country
3683	2014-06-30 00:00:00	3.0	2.50	1460	1613	2.0	0	0	3	1180	280	2007	0	6710 Alonzo Ave NW	Seattle	WA 98117	USA
4411	2014-05-22 00:00:00	5.0	2.25	2000	7900	1.0	0	0	4	1300	700	1986	0	3202 S 194th St	SeaTac	WA 98188	USA
2584	2014-06-16 00:00:00	3.0	3.25	2940	5432	3.0	0	3	4	2440	500	1978	2000	150 Highland Dr	Seattle	WA 98109	USA
69	2014-05-04 00:00:00	3.0	2.50	2200	7350	1.0	0	0	5	1570	630	1988	0	13420 SE 182nd St	Renton	WA 98058	USA
1844	2014-06-04 00:00:00	3.0	2.50	1720	8755	1.0	0	0	3	1000	720	1983	2009	31607 45th Pl SW	Federal Way	WA 98023	USA

	price
1898	685000.0

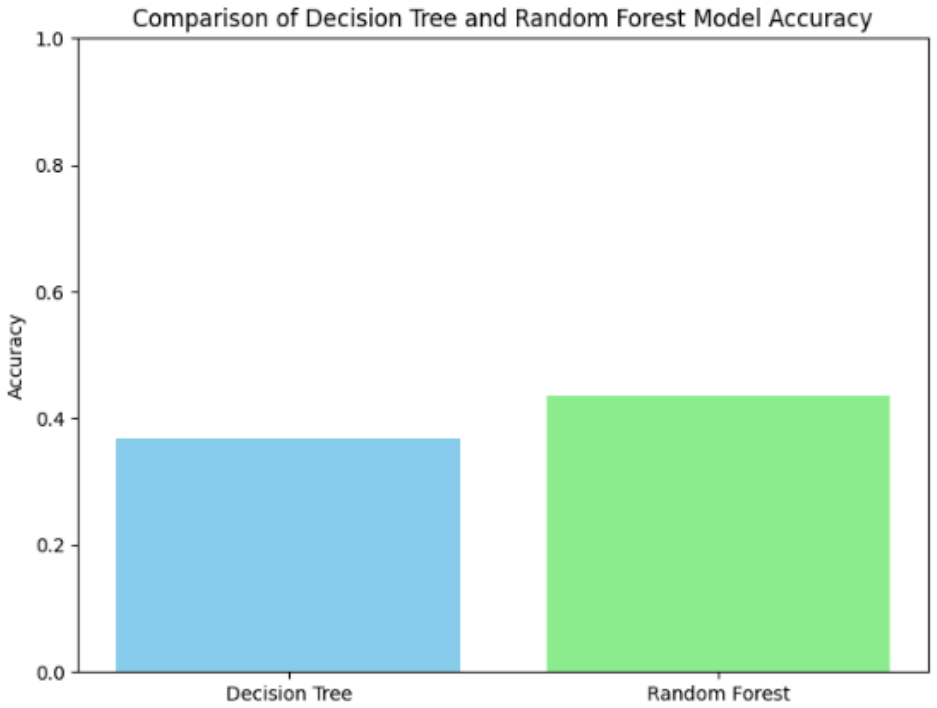
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
import pandas as pd

# Discretize the target variable 'price' into bins
y_train_classified = pd.qcut(y_train, q=5, labels=False, duplicates='drop')
y_test_classified = pd.qcut(y_test, q=y_train_classified.nunique(), labels=False, duplicates='drop')

# Instantiate a DecisionTreeClassifier object.
dt_classifier = DecisionTreeClassifier(random_state=42)

# Fit the decision tree model to the training data with the discretized target variable.
dt_classifier.fit(X_train, y_train_classified)

# Make predictions on the testing data.
y_pred_dt = dt_classifier.predict(X_test)

# Calculate the accuracy of the model.
accuracy_dt = accuracy_score(y_test_classified, y_pred_dt)

# Print the calculated accuracy score.
print(f"Decision Tree Accuracy: {accuracy_dt}")
```

```
Decision Tree Accuracy: 0.3695652173913043
```

```python
from sklearn.ensemble import RandomForestClassifier

# Instantiate a RandomForestClassifier object with random_state=42.
rf_classifier = RandomForestClassifier(random_state=42)

# Fit the Random Forest model to the training data using the discretized target variable.
rf_classifier.fit(X_train, y_train_classified)

# Make predictions on the testing data using the fitted model.
y_pred_rf = rf_classifier.predict(X_test)

# Calculate the accuracy of the Random Forest model by comparing the predictions with the discretized test target variable.
accuracy_rf = accuracy_score(y_test_classified, y_pred_rf)

# Print the calculated accuracy score.
print(f"Random Forest Accuracy: {accuracy_rf}")
```

```
Random Forest Accuracy: 0.43478260869565216
```

```python
import matplotlib.pyplot as plt

# Model names and their accuracies
models = ['Decision Tree', 'Random Forest']
accuracies = [accuracy_dt, accuracy_rf]

# Create a bar chart
plt.figure(figsize=(8, 6))
plt.bar(models, accuracies, color=['skyblue', 'lightgreen'])
plt.ylabel('Accuracy')
plt.title('Comparison of Decision Tree and Random Forest Model Accuracy')
plt.ylim(0, 1) # Set the y-axis limit from 0 to 1 for accuracy
plt.show()
```



```python
print(f"Decision Tree Accuracy: {accuracy_dt}")
print(f"Random Forest Accuracy: {accuracy_rf}")

if accuracy_rf > accuracy_dt:
    print("The Random Forest model performed better than the Decision Tree model based on accuracy.")
elif accuracy_dt > accuracy_rf:
    print("The Decision Tree model performed better than the Random Forest model based on accuracy.")
else:
    print("Both models performed equally well based on accuracy.")
```

```
Decision Tree Accuracy: 0.3695652173913043
Random Forest Accuracy: 0.43478260869565216
The Random Forest model performed better than the Decision Tree model based on accuracy.
```