Harshit Mishra(19BCE0799)

CSE3024 – WEB MINING

LAB ASSIGNMENT 11

Ques: The following are the basic steps involved in performing the random forest algorithm:

1. Pick N random records from the dataset.
2. Build a decision tree based on these N records.
3. Choose the number of trees you want in your algorithm and repeat steps 1 and 2.
4. In case of regression problem, for a new record, each tree in the forest predicts a value of Y. The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

Dataset Used: petrol_consumption.csv from Kaggle

Harshit Mishra(19BCE0799)

Procedure:

- We first import the dataset into our workspace using pandas.
- Then we define the set of dependent and independent attributes.
- We then import the random forest regressor from skleanrn.ensemble and train our model using the independent and dependent attributes.
- Next, we print the results of independent set as predicted by our regressor.
- Finally, we print all the evaluation metrics to check for the performance of our dataset.

We have not split the dataset into training set and test set as the number of entries in our dataset is only 48 and is relatively insufficient to split and then train with the remaining count of entries.

Code:

```
#Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#Importing the Dataset
dataset = pd.read_csv("petrol_consumption.csv")

#First few rows of our dataset
dataset.head(10)

#Checcking for null values
print(dataset.info())

#Set of independent and dependent attributes
X = dataset.iloc[:, 0:4].values
y = dataset.iloc[:, -1].values
```

```
#Training our Random Forest Regression Model
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators=200, random_state=0)
regressor.fit(X, y)

#Predictions by Regressor
y_pred = regressor.predict(X)

#Printing Mean Absolute Error
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y, y_pred)

#Printing Mean Absolute Error
from sklearn.metrics import mean_squared_error
mean_squared_error(y, y_pred)

#Printing Root Mean Squared Error
np.sqrt(mean_squared_error(y, y_pred))

#Printing Root Mean Sqaured Log Error
np.log(np.sqrt(mean_squared_error(y, y_pred)))

#Printing R-square value
from sklearn.metrics import r2_score
r2_score(y, y_pred)
```

Code Snippet and Explanation:

```
In [1]: #Importing Libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
```

Here we are importing the necessary libraries in our workspace. The pandas library will be used to import the dataset into out workspace. numpy library is used to convert arrays and list and reshape them according to the need of input parameters to a method. matplotlib is used to visualize our result, we will be using its pyplot sub-library to do the same.

```
In [2]: #Importing the Dataset
        dataset = pd.read_csv("petrol_consumption.csv")
```

Here we are importing the dataset into our workspace.

```
In [3]: #First few rows of our dataset
        dataset.head(10)
```

Out[3]:

	Petrol_tax	Average_income	Paved_Highways	Population_Driver_licence(%)	Petrol_Consumption
0	9.0	3571	1976	0.525	541
1	9.0	4092	1250	0.572	524
2	9.0	3865	1586	0.580	561
3	7.5	4870	2351	0.529	414
4	8.0	4399	431	0.544	410
5	10.0	5342	1333	0.571	457
6	8.0	5319	11868	0.451	344
7	8.0	5126	2138	0.553	467
8	8.0	4447	8577	0.529	464
9	7.0	4512	8507	0.552	498

```
In [4]: #Checcking for null values
        print(dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48 entries, 0 to 47
Data columns (total 5 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Petrol_tax                   48 non-null     float64
 1   Average_income               48 non-null     int64  
 2   Paved_Highways               48 non-null     int64  
 3   Population_Driver_licence(%)  48 non-null     float64
 4   Petrol_Consumption           48 non-null     int64  
dtypes: float64(2), int64(3)
memory usage: 2.0 KB
None
```

Here we are trying to understand the configuration of different attributes in our dataset. These configurations include checking for missing values and categorical inputs.

```
In [5]: #Set of independent and dependent attributes
        X = dataset.iloc[:, 0:4].values
        y = dataset.iloc[:, -1].values

In [6]: #Training our Random Forest Regression Model
        from sklearn.ensemble import RandomForestRegressor
        regressor = RandomForestRegressor(n_estimators=200, random_state=0)
        regressor.fit(X, y)

Out[6]: RandomForestRegressor(n_estimators=200, random_state=0)
```

Here we have defined our set of dependent and independent attributes. We have then trained our Random Forest Regressor using the above datasets. As explained earlier we have not split our dataset into training se and test set due to insufficient data entries.

The n_estimators here indicate the number of decision trees that we are using to train our random forest regressor. Hence we are using 200 decision trees for prediction. For final value we have used the average value of each decision tree to find the final consumption of petrol of a particular region.

```
In [7]: #Predictions by Regressor
        y_pred = regressor.predict(X)

In [8]: #Printing Mean Absolute Error
        from sklearn.metrics import mean_absolute_error
        mean_absolute_error(y, y_pred)

Out[8]: 16.542083333333327
```

Here we have predicted the results of X set as predicted by our random forest regressor. We have also printed the mean absolute error value.

```
In [9]: #Printing Mean Absolute Error
        from sklearn.metrics import mean_squared_error
        mean_squared_error(y, y_pred)

Out[9]: 676.4954427083334

In [10]: #Printing Root Mean Squared Error
         np.sqrt(mean_squared_error(y, y_pred))

Out[10]: 26.00952599930136

In [11]: #Printing Root Mean Sqaured Log Error
         np.log(np.sqrt(mean_squared_error(y, y_pred)))

Out[11]: 3.258462855507552

In [12]: #Printing R-square value
         from sklearn.metrics import r2_score
         r2_score(y, y_pred)

Out[12]: 0.9448102799874128
```

These are the evaluation metrics that we have used to evaluate the performance of our random forest regressor.

Results and Conclusion:

- **Mean Absolute Error:** 16.542
- **Mean Squared Error:** 676.495
- **Root Mean Squared Error:** 26.009
- **Root Mean Squared Log Error:** 3.258
- **R^2 Value:** 0.944

Harshit Mishra(19BCE0799)

Ques: The following are the basic steps involved in performing the random forest algorithm:

5. Pick N random records from the dataset.
6. Build a decision tree based on these N records.
7. Choose the number of trees you want in your algorithm and repeat steps 1 and 2.
8. In case of regression problem, for a new record, each tree in the forest predicts a value of Y. The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

Dataset Used: bill_authentication.csv from Kaggle

Procedure:

- We first import the dataset into our workspace using pandas.
- Then we define the set of dependent and independent attributes.
- We split the dataset in training set and test set with 20% of data in test set and remaining 80% of them being in training set.
- We then import the random forest regressor from skleanrn.ensemble and train our model using the independent and dependent attributes.
- Next, we print the results of independent set as predicted by our classifier.
- Finally, we print all the evaluation metrics to check for the performance of our dataset.

Harshit Mishra(19BCE0799)

Code:

```
#Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#Importing the Dataset
dataset = pd.read_csv("bill_authentication.csv")

#First few rows of our dataset
dataset.head(10)

#Set of independent and dependent attributes
X = dataset.iloc[:, 0:4].values
y = dataset.iloc[:, -1].values

#Splitting the dataset into training set and test set
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=0)

#Training our Random Forest Regression Model
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators=200, random_state=0)
classifier.fit(X_train,y_train)

#Predictions by Regressor
y_pred = classifier.predict(X)

#Printing Mean Absolute Error
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y, y_pred)

#Printing Mean Absolute Error
from sklearn.metrics import mean_squared_error
mean_squared_error(y, y_pred)

#Printing Root Mean Squared Error
np.sqrt(mean_squared_error(y, y_pred))

#Printing Root Mean Sqaured Log Error
np.log(np.sqrt(mean_squared_error(y, y_pred)))

#Printing the Confusion Matrix
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, y_pred)
```

#Printing R-square value
from sklearn.metrics import r2_score
r2_score(y, y_pred)

Code Snippet and Explanation:

```
In [1]: #Importing Libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
```

Here we are importing the necessary libraries in our workspace. The pandas library will be used to import the dataset into out workspace. numpy library is used to convert arrays and list and reshape them according to the need of input parameters to a method. matplotlib is used to visualize our result, we will be using its pyplot sub-library to do the same.

```
In [2]: #Importing the Dataset
        dataset = pd.read_csv("bill_authentication.csv")
```

Here we are importing the dataset into our workspace.

```
In [3]: #Printing the first few rows of the dataset
        dataset.head()
```

Out[3]:

	Variance	Skewness	Curtosis	Entropy	Class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	1

```
In [4]: #Checcking for null values
        print(dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48 entries, 0 to 47
Data columns (total 5 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Petrol_tax                    48 non-null     float64
 1   Average_income                48 non-null     int64  
 2   Paved_Highways                48 non-null     int64  
 3   Population_Driver_licence(%)  48 non-null     float64
 4   Petrol_Consumption            48 non-null     int64  
dtypes: float64(2), int64(3)
memory usage: 2.0 KB
None
```

Here we are trying to understand the configuration of different attributes in our dataset. These configurations include checking for missing values and categorical inputs.

```
In [4]: #Defining set of Dependent and Independent Set
        X = dataset.iloc[:, 0:4].values
        y = dataset.iloc[:, -1].values
```

```
In [5]: #Splitting the dataset into training set and test set
        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, rando
```

Here we are defining our set of dependent and independent variables as X and y respectively. Next we are splitting our dataset as we have mentioned in the procedure.

```
In [6]: #Training our Random Forest Regression Model
        from sklearn.ensemble import RandomForestClassifier
        classifier = RandomForestClassifier(n_estimators=200, random_state=0)
        classifier.fit(X_train,y_train)

Out[6]: RandomForestClassifier(n_estimators=200, random_state=0)

In [7]: #Predictions by Regressor
        y_pred = classifier.predict(X_test)
```

Here we are defining our classifier and initializing the classifier object with RandomForestClassifier. The n_estimators here indicate the number of decision trees that we are using to train our random forest regressor. Hence, we are using 200 decision trees for prediction. For final value we have used the average value of each decision tree to find the final consumption of petrol of a particular region.

Next, we are also predicting the test set results using the classifiers .predict method.

```
In [8]: #Printing Mean Absolute Error
        from sklearn.metrics import mean_absolute_error
        mean_absolute_error(y_test, y_pred)

Out[8]: 0.04727272727272727

In [9]: #Printing Mean Squared Error
        from sklearn.metrics import mean_squared_error
        mean_squared_error(y_test, y_pred)

Out[9]: 0.04727272727272727

In [10]: #Printing Root Mean Squared Error
         np.sqrt(mean_squared_error(y_test, y_pred))

Out[10]: 0.2174229226018436

In [11]: #Printing Root Mean Squared Log Error
         np.log(np.sqrt(mean_squared_error(y_test, y_pred)))

Out[11]: -1.5259108701025172

In [12]: #Printing R-square value
         from sklearn.metrics import r2_score
         r2_score(y_test, y_pred)

Out[12]: 0.8084753026893818
```

Here we are evaluating the various evaluation metrics from sklearn.metrics and are evaluating our model's performance.

```
In [13]: #Printing the Confusion Matrix
         from sklearn.metrics import confusion_matrix
         confusion_matrix(y_test, y_pred)

Out[13]: array([[148,   5],
                [  8, 114]], dtype=int64)
```

Here we are printing the confusion matrix of our model's performance on test set.

Harshit Mishra(19BCE0799)

Results and Conclusion:

- **Mean Absolute Error:** 0.04727
- **Mean Squared Error:** 0.04727
- **Root Mean Squared Error:** 0.21742
- **Root Mean Squared Log Error:** -1.52591
- **R^2 Value:** 0.80847

- **True Positives:** 148
- **True Negatives:** 114
- **False Positives:** 5
- **False Negatives:** 8