Random Forest

Machine Learning Technique

Imesh Chamara
Index number: s14216
Degree: Compuational Chemistry
2018s116744@stu.cmb.ac.lk

Introduction

Random forest is a supervised learning algorithm which use both classification and regression. Random forests create decision trees on randomly selected data samples, gets predictions from each tree and select the best solution. Random forests are used in various kind of applications. Image classification, feature selection and recommendation engines can be given as the examples.

There are advantages and disadvantages of using random forests. It is a highly accurate and robust method because of the number of decision trees participating in the process. The algorithm can be used both classification and regression problems. Missing values also handled by random forests. When considering the disadvantages, it can be considered as a time consuming process. Random forests are difficult to build when compared to decision trees. More resources are needed for the computation as well.

Data set

This dataset has details of 1000 users from different backgrounds and whether or not they buy a bike. This data can be used for prediction models using Machine Learning Algorithms. There are some NA values injected in the dataset. Use this dataset for Data Cleaning, Exploration and Visualization.

GitHub : https://raw.githubusercontent.com/iMeshCMR/Machine_Learning/main/Bike_Buyers%20_data.csv

Kaggle: https://www.kaggle.com/heeraldedhia/bike-buyers

Problem

It needs to predict how customer's buying decision of bikes depends on marital status, gender, education, occupation and home owner.

Approach

A matching data set was selected for the random forest machine learning technique and then the random forest was modeled. After that accuracy of the decision tree and random forest was calculated and the predictions was taken based on that values.

Steps

The data set was uploaded into the GitHub and the coding was done for the random forest using Goole Colab. Finally accuracy of the decision tree and the random forest were calculated.

Bike Buyers

This dataset has details of 1000 users from different backgrounds and whether or not they buy a bike. This data can be used for prediction models using Machine Learning Algorithms.

Columns -

ID Marital Status Gender Income Children Education Occupation Home Owner Cars Commute Distance Region Age Purchased Bike

Importing libraries

```
[ ] import numpy as np
    import pandas as pd
    from sklearn import preprocessing
    import matplotlib.pyplot as plt
    import seaborn as sns
    import warnings
    warnings.simplefilter(action='ignore')
```

Importing the CSV file and readin the file

```
[ ] datafile =  pd.read_csv('https://raw.githubusercontent.com/iMeshCMR/Machine_Learning/main/Bike_Buyers%20_data.csv', sep=",")
    print(datafile.head())
```

```
[ ]       ID Marital_Status  Gender  ...   Region   Age Purchased_Bike
    0  12496        Married  Female  ...   Europe  42.0             No
    1  24107        Married    Male  ...   Europe  43.0             No
    2  14177        Married    Male  ...   Europe  60.0             No
    3  24381         Single  Female  ...  Pacific  41.0            Yes
    4  25597         Single    Male  ...   Europe  36.0            Yes

    [5 rows x 13 columns]
```

Decribing the data file

```
datafile.describe()
```

	ID	Income	Children	Cars	Age
count	1000.000000	1000.00000	992.000000	991.000000	992.000000
mean	19965.992000	56210.00000	1.910282	1.455096	44.181452
std	5347.333948	30995.10601	1.626910	1.121755	11.362007
min	11000.000000	10000.00000	0.000000	0.000000	25.000000
25%	15290.750000	30000.00000	0.000000	1.000000	35.000000
50%	19744.000000	60000.00000	2.000000	1.000000	43.000000
75%	24470.750000	70000.00000	3.000000	2.000000	52.000000
max	29447.000000	170000.00000	5.000000	4.000000	89.000000

View statistical properties of the datafile Checking whether there is any missing values in the datafile

```
[ ] datafile.isnull().sum()
```

```
    ID                  0
    Marital_Status      0
    Gender              0
    Income              0
    Children            8
    Education           0
    Occupation          0
    Home_Owner          0
    Cars                9
    Commute Distance    0
    Region              0
    Age                 8
    Purchased_Bike      0
    dtype: int64
```

Checking all the column names

```
[ ] datafile.columns
```

```
    Index(['ID', 'Marital_Status', 'Gender', 'Income', 'Children', 'Education',
           'Occupation', 'Home_Owner', 'Cars', 'Commute Distance', 'Region', 'Age',
           'Purchased_Bike'],
          dtype='object')
```

Drop useless data

```
[ ] #Because there are some string values that cant be cateorised
    datafile=datafile.drop(['ID','Income','Children','Cars','Commute Distance','Region','Age'], axis=1)
    datafile.head()
```

	Marital_Status	Gender	Education	Occupation	Home_Owner	Purchased_Bike
0	Married	Female	Bachelors	Skilled Manual	Yes	No
1	Married	Male	Partial College	Clerical	Yes	No
2	Married	Male	Partial College	Professional	No	No
3	Single	Female	Bachelors	Professional	Yes	Yes
4	Single	Male	Bachelors	Clerical	No	Yes

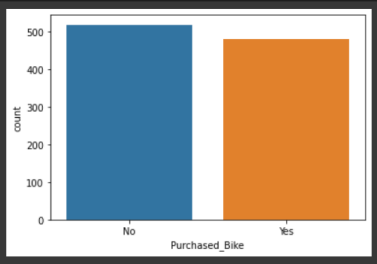
View categorical varibales

```
[ ] Categorical = [var for var in datafile.columns if datafile[var].dtype=='O']
    datafile[Categorical].head()
```

	Marital_Status	Gender	Education	Occupation	Home_Owner	Purchased_Bike
0	Married	Female	Bachelors	Skilled Manual	Yes	No
1	Married	Male	Partial College	Clerical	Yes	No
2	Married	Male	Partial College	Professional	No	No
3	Single	Female	Bachelors	Professional	Yes	Yes
4	Single	Male	Bachelors	Clerical	No	Yes

Checking the distribution of the data set

```
#Here dependent data will mainly focused
datafile.Purchased_Bike.value_counts()
sns.countplot(x="Purchased_Bike", data=datafile)
plt.show()
```



Checking all the unique categories in Education and Occupation before turn into numerical

```
[ ] datafile.Education.unique()
```

```
array(['Bachelors', 'Partial College', 'High School',
       'Partial High School', 'Graduate Degree'], dtype=object)
```

```
[ ] datafile.Occupation.unique()
```

```
array(['Skilled Manual', 'Clerical', 'Professional', 'Manual',
       'Management'], dtype=object)
```

Assining vlues in the Purchased_Bike column to numerical values

```
[ ] scale_mapper_Education = {'Bachelors':1, 'Partial College':2, 'High School':3,
        'Partial High School':4, 'Graduate Degree':5}
    scale_mapper_Occupation = {'Skilled Manual':1, 'Clerical':2, 'Professional':3, 'Manual':4,
        'Management':5}
    scale_mapper_Marital_Status = {"Married":1,"Single":0}
    scale_mapper_Gender  = {"Male":1,"Female":0}
    scale_mapper_Home_Owner = {"Yes":1,"No":0}
    scale_mapper_Purchased_Bike = {"Yes":1,"No":0}

    datafile["Education_N"] = datafile["Education"].replace(scale_mapper_Education)
    datafile["Occupation_N"] = datafile["Occupation"].replace(scale_mapper_Occupation)
    datafile["Marital_Status_N"] = datafile["Marital_Status"].replace(scale_mapper_Marital_Status)
    datafile["Gender_N"] = datafile["Gender"].replace(scale_mapper_Gender)
    datafile["Home_Owner_N"] = datafile["Home_Owner"].replace(scale_mapper_Home_Owner)
    datafile["Purchased_Bike_N"] = datafile["Purchased_Bike"].replace(scale_mapper_Purchased_Bike)

    print(datafile.head())
```

```
  Marital_Status  Gender  ... Home_Owner_N Purchased_Bike_N
0        Married  Female  ...            1               0
1        Married    Male  ...            1               0
2        Married    Male  ...            0               0
3         Single  Female  ...            1               1
4         Single    Male  ...            0               1

[5 rows x 12 columns]
```

```
[ ] datafile=datafile.drop(["Education"], axis=1)
    datafile=datafile.drop(["Occupation"], axis=1)
    datafile=datafile.drop(["Marital_Status"], axis=1)
    datafile=datafile.drop(["Gender"], axis=1)
    datafile=datafile.drop(["Home_Owner"], axis=1)
    datafile=datafile.drop(["Purchased_Bike"], axis=1)
    datafile.head()
```

	Education_N	Occupation_N	Marital_Status_N	Gender_N	Home_Owner_N	Purchased_Bike_N
0	1	1	1	0	1	0
1	2	2	1	1	1	0
2	2	3	1	1	0	0
3	1	3	0	0	1	1
4	1	2	0	1	0	1

Splitting data set to train and test the accuracy later

```
from sklearn.model_selection import train_test_split
y = datafile.Purchased_Bike_N.values
x_data = datafile.drop(['Purchased_Bike_N'], axis = 1)

x_train, x_test, y_train, y_test = train_test_split(x_data,y,test_size = 0.2,random_state=0) #20% for the test size
x_train
```

	Education_N	Occupation_N	Marital_Status_N	Gender_N	Home_Owner_N
496	1	1	0	0	1
558	1	1	1	0	1
784	3	4	0	0	1
239	3	4	0	0	0
578	1	5	1	1	1
...	...	...	...	...	...
835	1	1	0	0	1
192	1	5	0	0	1
629	5	1	1	0	1
559	1	5	0	0	1
684	1	1	0	0	0

990 rows × 5 columns

Inferences

Decision tree

```
[ ] from sklearn.tree import DecisionTreeClassifier
    DisTrClass = DecisionTreeClassifier()
    DisTrClass.fit(x_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

Acuracy od Decision tree

```
[ ] DisTrClass.score(x_test, y_test)*100
```

```
57.49999999999999
```

Random forest

```
[ ] from sklearn.ensemble import RandomForestClassifier
    RandForClass = RandomForestClassifier()
    RandForClass.fit(x_train, y_train)
```

```
[ ] RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                           criterion='gini', max_depth=None, max_features='auto',
                           max_leaf_nodes=None, max_samples=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=100,
                           n_jobs=None, oob_score=False, random_state=None,
                           verbose=0, warm_start=False)
```

Checking acuracy of Random Forest

```
[ ] RandForClass.score(x_test, y_test)*100

    62.5
```

Develop confusion matrix

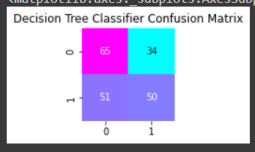
```
[ ] y_DisTrClass = DisTrClass.predict(x_test)
    y_RandForClass = RandForClass.predict(x_test)
```

```
[ ] from sklearn.metrics import confusion_matrix

    CM_DisTrClass = confusion_matrix(y_test,y_DisTrClass)
    CM_RandForClass = confusion_matrix(y_test,y_RandForClass)
```

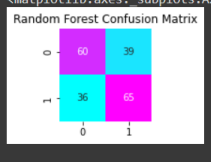
```
[ ] plt.subplot(2,3,1)
    plt.title("Decision Tree Classifier Confusion Matrix")
    sns.heatmap(CM_DisTrClass,annot=True,cmap="cool",fmt="d",cbar=False)
```

```
    <matplotlib.axes._subplots.AxesSubplot at 0x7f1238839110>
```




```
[ ] plt.subplot(2,3,2)
    plt.title("Random Forest Confusion Matrix")
    sns.heatmap(CM_RandForClass,annot=True,cmap="cool",fmt="d",cbar=False)

    <matplotlib.axes._subplots.AxesSubplot at 0x7f12388ad890>
```



Conclusion

When considering accuracy values of the decision tree and the random forest, the accuracy of the random forest (62.5%) is higher than the accuracy of the decision tree(57.4999999999999%). This is because the random forest classifier will handle the missing values and maintain the accuracy of a large proportion of data. But this accuracy values are low. This can be because the data set contain small number of data. If a large data set could be used for this this problem can be solved.

Using this model it can be concluded that customer's buying decision of bikes depends on marital status, gender, education, occupation and home owner. By drawing further more graphs we can conclude which parameter is mainly interfere with the customer's buying decision of bikes.

Colab link

https://colab.research.google.com/drive/15vWEcC0a6pXqaftyf-08-Q2xTQzp_CIH?usp=sharing

References

(1) Introduction to Random Forest in Machine Learning https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/.