Eternaltek

Day 19-100 of Data Science

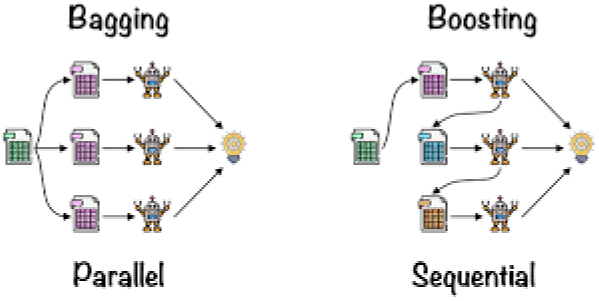
Random Forest Algorithm

- A random forest is an ensemble learning method that combines the predictions from multiple decision trees to produce a more accurate and stable prediction.
- It is a type of supervised learning algorithm that can be used for both classification and regression tasks.
- Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.
- The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Types of Ensemble Methods

There are various types of ensemble learning methods, including:

- Bagging (Bootstrap Aggregating): This method involves training multiple models on random subsets of the training data. The predictions from the individual models are then combined, typically by averaging.
- Boosting: This method involves training a sequence of models, where each subsequent model focuses on the errors made by the previous model. The predictions are combined using a weighted voting scheme.



Why use Random Forest?

- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.

Implementation

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
import warnings

from sklearn.preprocessing import LabelEncoder
from sklearn.impute import KNNImputer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import f1_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score

warnings.filterwarnings('ignore')
```

In [5]:

```
data = 'car_evaluation.csv'

df = pd.read_csv(data)
```

In [6]:

```
df.head()
```

Out[6]:

	vhigh	vhigh.1	2	2.1	small	low	unacc
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc

In [7]:

```
# view dimensions of dataset

df.shape
```

Out[7]:

```
(1727, 7)
```

In [8]:

```
# preview the dataset

df.head()
```

Out[8]:

	vhigh	vhigh.1	2	2.1	small	low	unacc
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc

In [9]:

```
#Rename column names
#We can see that the dataset does not have proper column names.
#The columns are merely labelled as 0,1,2.... and so on.
#We should give proper names to the columns. I will do it as follows:-

col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']


df.columns = col_names

col_names
```

Out[9]:

```
['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']
```

In [10]:

```
# let's again preview the dataset

df.head()
```

Out[10]:

	buying	maint	doors	persons	lug_boot	safety	class
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc

In [11]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1727 entries, 0 to 1726
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   buying    1727 non-null   object
 1   maint     1727 non-null   object
 2   doors     1727 non-null   object
 3   persons   1727 non-null   object
 4   lug_boot  1727 non-null   object
 5   safety    1727 non-null   object
 6   class     1727 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

In [12]:

```
#Frequency distribution of values in variables
#Now, I will check the frequency counts of categorical variables.

col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']


for col in col_names:

    print(df[col].value_counts())
```

```
buying
high     432
med      432
low      432
vhigh    431
Name: count, dtype: int64
maint
high     432
med      432
low      432
vhigh    431
Name: count, dtype: int64
doors
3        432
4        432
5more    432
2        431
Name: count, dtype: int64
persons
4       576
more    576
2       575
Name: count, dtype: int64
lug_boot
med      576
big      576
small    575
Name: count, dtype: int64
safety
med     576
high    576
low     575
Name: count, dtype: int64
class
unacc    1209
acc       384
good       69
vgood      65
Name: count, dtype: int64
```

In [13]:

```
df['class'].value_counts()
```

Out[13]:

```
class
unacc    1209
acc       384
good       69
vgood      65
Name: count, dtype: int64
```

In [14]:

```
# check missing values in variables

df.isnull().sum()
```

Out[14]:

```
buying      0
maint       0
doors       0
persons     0
lug_boot    0
safety      0
class       0
dtype: int64
```

In [15]:

```
X = df.drop(['class'], axis=1)

y = df['class']
```

In [16]:

```
# split data into training and testing sets

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_sta
```

In [17]:

```
# check the shape of X_train and X_test

X_train.shape, X_test.shape
```

Out[17]:

```
((1157, 6), (570, 6))
```

In [18]:

```
# check data types in X_train

X_train.dtypes
```

Out[18]:

```
buying      object
maint       object
doors       object
persons     object
lug_boot    object
safety      object
dtype: object
```

In [19]:

```
X_train.head()
```

Out[19]:

	buying	maint	doors	persons	lug_boot	safety
83	vhigh	vhigh	5more	2	med	low
48	vhigh	vhigh	3	more	med	med
468	high	vhigh	3	4	small	med
155	vhigh	high	3	more	med	low
1043	med	high	4	more	small	low

In [23]:

```
# import category encoders
#!pip install --upgrade category_encoders
import category_encoders as ce
```

In [24]:
```
# encode categorical variables with ordinal encoding

encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'doors', 'persons', 'lug_boot', '


X_train = encoder.fit_transform(X_train)

X_test = encoder.transform(X_test)
```

In [25]:
```
X_train.head()
```

Out[25]:

	buying	maint	doors	persons	lug_boot	safety
83	1	1	1	1	1	1
48	1	1	2	2	1	2
468	2	1	2	3	2	2
155	1	2	2	2	1	1
1043	3	2	3	2	2	1

In [26]:
```
X_test.head()
```

Out[26]:

	buying	maint	doors	persons	lug_boot	safety
599	2	2	3	1	3	1
932	3	1	3	3	3	1
628	2	2	1	1	3	3
1497	4	2	1	3	1	2
1262	3	4	3	2	1	1

In [34]:
```
# import Random Forest classifier

from sklearn.ensemble import RandomForestClassifier

# instantiate the classifier

rfc = RandomForestClassifier(random_state=60)

# fit the model

rfc.fit(X_train, y_train)
```

Out[34]:
```
▾ RandomForestClassifier
RandomForestClassifier(random_state=60)
```

In [35]:
```
# Predict the Test set results

y_pred = rfc.predict(X_test)
```

In [36]:

```
# Check accuracy score

from sklearn.metrics import accuracy_score

print('Model accuracy score with 10 decision-trees : {0:0.4f}'. format(accuracy_score(
```

```
Model accuracy score with 10 decision-trees : 0.9632
```

In [ ]:

Follow us on social media

Linkedin: https://www.linkedin.com/company/eternaltek/about/?viewAsMember=true

Medium: https://medium.com/@eternaltek.info

WhatsApp Channel: https://whatsapp.com/channel/0029Va5onCbDjiOTi6D1vU36

Github: https://github.com/Vamsi-2203

In [ ]: