Untitled

November 17, 2018

Applying ML Classication algorithms on Car Evaluation Data Set and getting inferences from the data

0.0.1 Data Source

Kaggle-Car evalation data set.

Link-kaggle datasets download -d elikplim/car-evaluation-data-set ### Data Description Car Evaluation Database was derived from a simple hierarchical decision model originally developed for the demonstration of DEX, M. Bohanec, V. Rajkovic: Expert system for decision making. Sistemica 1(1), pp. 145-157, 1990.). The model evaluates cars according to the following concept structure:

CAR car acceptability . PRICE overall price . . buying buying price . . maint price of the maintenance . TECH technical characteristics . . COMFORT comfort . . . doors number of doors . . . persons capacity in terms of persons to carry . . . lug_boot the size of luggage boot . . safety estimated safety of the car

Input attributes are printed in lowercase. Besides the target concept (CAR), the model includes three intermediate concepts: PRICE, TECH, COMFORT. Every concept is in the original model related to its lower level descendants by a set of examples (for these examples sets see [Web Link]).

The Car Evaluation Database contains examples with the structural information removed, i.e., directly relates CAR to the six input attributes: buying, maint, doors, persons, lug_boot, safety.

Because of known underlying concept structure, this database may be particularly useful for testing constructive induction and structure discovery methods.

Attribute Information:

Class Values:

unacc, acc, good, vgood

Attributes:

buying: vhigh, high, med, low.

maint: vhigh, high, med, low.

doors: 2, 3, 4, 5more.

persons: 2, 4, more.

lug_boot: small, med, big. safety: low, med, high.

0.1 Importing Libraries

```
In [3]: import pandas as pd
        import numpy as np

        import matplotlib.pyplot as plt
```

```
# disable warnings
import warnings

warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn.ensemble import RandomForestClassifier
```

0.2 Loading data set

```
In [4]: filename = 'car_evaluation.csv'
        data=pd.read_csv(filename)
```

```
In [5]: # Assign names to Columns
        data.columns = ['buying','maint','doors','persons','lug_boot','safety','classes']
```

0.3 Viewing Data

```
In [6]: data.head()
```

```
Out[6]:   buying  maint doors persons lug_boot safety classes
        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]: data.to_csv("test2.csv")
```

0.4 Information about data

```
In [8]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1727 entries, 0 to 1726
Data columns (total 7 columns):
buying      1727 non-null object
maint       1727 non-null object
doors       1727 non-null object
persons     1727 non-null object
lug_boot    1727 non-null object
safety      1727 non-null object
classes     1727 non-null object
dtypes: object(7)
memory usage: 94.5+ KB
```

```
In [9]: # Encode Data
        data.buying.replace(('vhigh','high','med','low'),(1,2,3,4), inplace=True)
```

```
data.maint.replace(('vhigh','high','med','low'),(1,2,3,4), inplace=True)
data.doors.replace(('2','3','4','5more'),(1,2,3,4), inplace=True)
data.persons.replace(('2','4','more'),(1,2,3), inplace=True)
data.lug_boot.replace(('small','med','big'),(1,2,3), inplace=True)
data.safety.replace(('low','med','high'),(1,2,3), inplace=True)
data.classes.replace(('unacc','acc','good','vgood'),(1,2,3,4), inplace=True)
```

In [ ]:

0.5 Preprocessing of data

In [10]:
```
data.head()
```

Out[10]:

	buying	maint	doors	persons	lug_boot	safety	classes
0	1	1	1	1	1	2	1
1	1	1	1	1	1	3	1
2	1	1	1	1	2	1	1
3	1	1	1	1	2	2	1
4	1	1	1	1	2	3	1

In [11]:
```
data.describe()
```

Out[11]:

	buying	maint	doors	persons	lug_boot	\
count	1727.000000	1727.000000	1727.000000	1727.000000	1727.000000	
mean	2.500869	2.500869	2.500869	2.000579	2.000579	
std	1.118098	1.118098	1.118098	0.816615	0.816615	
min	1.000000	1.000000	1.000000	1.000000	1.000000	
25%	2.000000	2.000000	2.000000	1.000000	1.000000	
50%	3.000000	3.000000	3.000000	2.000000	2.000000	
75%	3.500000	3.500000	3.500000	3.000000	3.000000	
max	4.000000	4.000000	4.000000	3.000000	3.000000	

	safety	classes
count	1727.000000	1727.000000
mean	2.000579	1.415171
std	0.816615	0.740847
min	1.000000	1.000000
25%	1.000000	1.000000
50%	2.000000	1.000000
75%	3.000000	2.000000
max	3.000000	4.000000

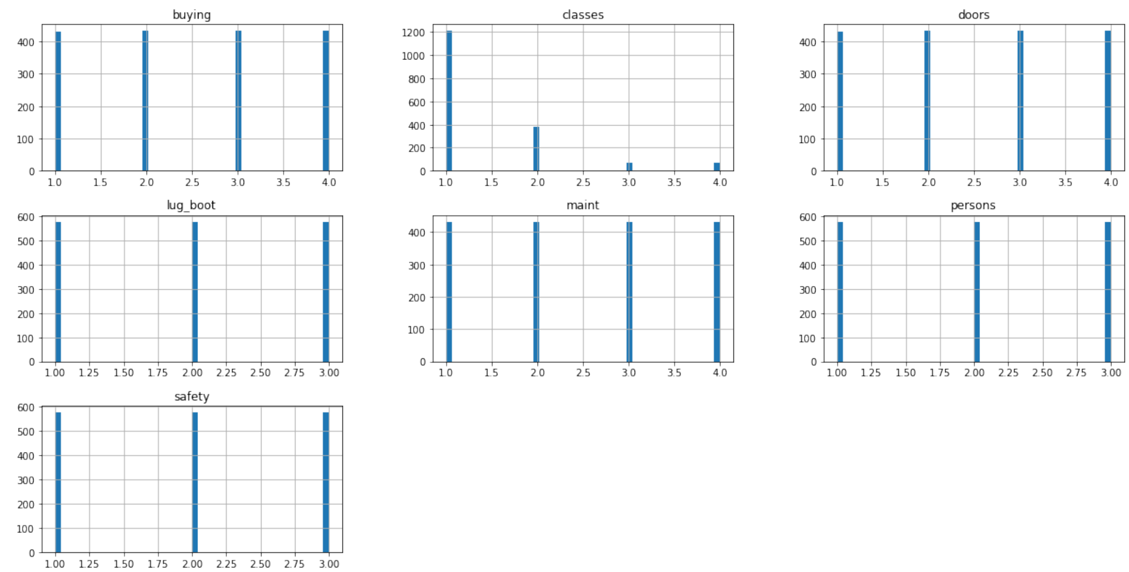
In [12]:
```
corr = data.corr()
corr
```

Out[12]:

	buying	maint	doors	persons	lug_boot	safety	classes
buying	1.000000	-0.001043	-0.001043	-0.000952	-0.000952	-0.000952	0.282488
maint	-0.001043	1.000000	-0.001043	-0.000952	-0.000952	-0.000952	0.232128
doors	-0.001043	-0.001043	1.000000	-0.000952	-0.000952	-0.000952	0.065662

```
persons  -0.000952 -0.000952 -0.000952  1.000000 -0.000869 -0.000869  0.341489
lug_boot -0.000952 -0.000952 -0.000952 -0.000869  1.000000 -0.000869  0.157617
safety   -0.000952 -0.000952 -0.000952 -0.000869 -0.000869  1.000000  0.439171
classes   0.282488  0.232128  0.065662  0.341489  0.157617  0.439171  1.000000
```

In [13]:
```python
data.hist(bins=50, figsize=(20, 10))
plt.show()
```



1 Classification

As we all know their are many classification algorithms that we can apply on our dataframe but which one is most suitable? Thus to solve the above query we compare different classification models on the basis of:- 1. Accuracy How much accuartly the classifier is able to classify the given data 2. Speed Time taken to construct the model (training time) Time taken to use the model (classification/prediction time) 3. Robustness handling noise and missing values 4. Scalability efficiency in disk-resident databases 5. Interpretability

Receiver Operating Characteristics Curevs: ROC curves also plays a very essential role in model selection. These curves shows the trade-off between the true positive rate and the false positive rate and the area under the curve is a measure of the accuracy of the model.

1.0.1 Ensemble Models:(Technique to improve classification accuracy)

To increase accuracy we combine a series of models with the aim of creating a improved model. Methods for constructing an Ensemble Classifier 1. By manipulating the training set: Bagging and Bosting 2. By manipulating the input features: Random forest 3. By manipulating the class labels. 4. By manipulating the learning algorithm. Why are ensemble models better?

Random forest: Thus here we use Random forest classification model and compare the output with the decision tree.

1.1 1. Decision Tree Algorithm

In [14]:
```
from sklearn.tree import DecisionTreeClassifier
car = data.values
X,y = car[:,:6], car[:,6]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_sta
clf_gini = DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=3, r
clf_gini.fit(X_train, y_train)
```

Out[14]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=3,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=5, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=100,
            splitter='best')
```

In [15]:
```
clf_gini.score(X_test,y_test)
```

Out[15]:
```
0.7822736030828517
```

In [16]:
```
y_pred = clf_gini.predict(X_test)
```

In [17]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
print("Confusion Matrix: ",confusion_matrix(y_test, y_pred))
print ("Accuracy : ",accuracy_score(y_test,y_pred)*100)
print("Report : ",classification_report(y_test, y_pred))
```

```
Confusion Matrix:  [[357  15   0   0]
 [ 56  49   0   0]
 [  0  22   0   0]
 [  0  20   0   0]]
Accuracy :  78.22736030828517
Report :               precision    recall  f1-score   support

          1       0.86      0.96      0.91       372
          2       0.46      0.47      0.46       105
          3       0.00      0.00      0.00        22
          4       0.00      0.00      0.00        20

avg / total       0.71      0.78      0.75       519
```

In [ ]:

1.2 2. Random Forest Algorithm

In [18]:
```
car = data.values
X,y = car[:,:6], car[:,6]
```

```
X,y = X.astype(int), y.astype(int)
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=0)
```

In [19]:
```
>>> clf = RandomForestClassifier(n_estimators=500)

>>> clf.fit(X_train,y_train)
```

Out[19]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', 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, n_estimators=500, n_jobs=1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [20]:
```
clf.score(X_test,y_test)

y_pred = clf.predict(X_test)
```

In [21]:
```
print("Confusion Matrix: ",confusion_matrix(y_test, y_pred))
print ("Accuracy : ",accuracy_score(y_test,y_pred)*100)
print("Report : ",classification_report(y_test, y_pred))
```

```
Confusion Matrix:  [[358   0   0   0]
 [  4 111   1   0]
 [  0   2  16   3]
 [  0   3   0  21]]
Accuracy :  97.495183044316
Report :               precision    recall  f1-score   support

          1       0.99      1.00      0.99       358
          2       0.96      0.96      0.96       116
          3       0.94      0.76      0.84        21
          4       0.88      0.88      0.88        24

avg / total       0.97      0.97      0.97       519
```

Thus we can observe that we are obtaining accuracy of 97.30% using random forest algorithm.

1.2.1 MinMax Scaling

In [22]:
```
from sklearn import preprocessing
scaler = preprocessing.MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.fit_transform(X_test)
```

In [23]:
```
clf.fit(X_train_scaled,y_train)
clf.score(X_test_scaled,y_test)
clf.score(X_test_scaled,y_test)
```

Out[23]: 0.976878612716763

In [24]:
```
y_pred = clf.predict(X_test_scaled)
print("Confusion Matrix: ",confusion_matrix(y_test, y_pred))
print ("Accuracy : ",accuracy_score(y_test,y_pred)*100)
print("Report : ",classification_report(y_test, y_pred))
```

```
Confusion Matrix:  [[358   0   0   0]
 [  3 112   1   0]
 [  0   1  16   4]
 [  0   3   0  21]]
Accuracy :  97.6878612716763
Report :               precision    recall  f1-score   support

          1       0.99      1.00      1.00       358
          2       0.97      0.97      0.97       116
          3       0.94      0.76      0.84        21
          4       0.84      0.88      0.86        24

avg / total       0.98      0.98      0.98       519
```

As we can see that by minmax scaling the accuracy of ourdata set is increased

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: