INTERNSHIP PROJECT REPORT

Car Model Acceptance Using AI-ML Models

Name of Project: CAR MODEL ACCEPTANCE USING AI – ML MODELS

……
……..

Name of Intern: Ngawang Choega

……
……..

Intern's Ph. no.: 7428708711

……
……..

Intern's Add.: H.No-3,Monastery Market,Budh Vihar,near ISBT,Delhi-110054

……
……..

Name of College: Jaypee University of Information Technology

……
……..

College's Add.: Wakhnaghat,Himachal Pradesh

……..……
……..

INFORMATION TECHNOLOGY

Branch:

……
……

Project Submission Date: 13/07/2022……..

Table of content

DECLARATION

I hereby declare that this project has been done by me under the supervision of **Mr Gagan Preet Singh,** Affiliation**,** Hex N Bit. I also declare that neither this project nor any part of this project has been submitted elsewhere for the award of any degree or diploma.

Supervised by:

(Mr Gagan Preet Singh)

Mentor

Hex N Bit

Submitted by:

(Ngawang Choega)

Jaypee University of Information Technology

CERTIFICATE

This is to certify that the work which is being presented in the project report titled "CAR MODEL ACCEPTANCE USING AI – ML MODELS" in partial fulfilment of the requirements for the award of the certificate of Summer Training at Hex N Bit, is an authentic record of work carried out by "Prakhar Srivastava" during the period from June 2022 to July 2022 under the supervision of Mr Gagan Preet Singh, Mentor, Hex N Bit.

Ngawang Choega

The above statement made is correct to the best of my knowledge.

(**Mr Gagan Preet Singh**)

Mentor,

Hex N Bit

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to Almighty God for His divine blessing for making it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to my Supervisor **Mr Gagan Preet Singh**, Mentor, Hex N Bit. The deep Knowledge & keen interest of my supervisor in the field of "**Machine Learning**" has helped me immensely in carrying out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, and valuable advice, have made it possible to complete this project.

I would like to express my heartiest gratitude to **Mr Gagan Preet Singh,** Mentor, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patience of my parents.

Ngawang Choega

ABSTRACT

When an individual considers buying a car, there are many aspects that could affect his/her decision on which type of car he/she is interested in. Factors such as the price, regular maintenance, comfort, and safety are a few vital issues that we need to consider. A car evaluation database is very important structural information that someone should look at for the features and classification of each car could be helpful in our decision making. [1]

The goal of this report is particularly to recognize the decision making, identify the ways in recognizing the car's ultimate value with certain attributes to be met and apply this pattern to a whole set of patterns as a classifier between a good acceptable car from the unacceptable ones.

List of Software:

- Jupyter Notebook
- Weka 3.8.6
- Google Colab
- MS Excel
- Notepad

List of Hardware:

- MAC OS
- 8.0 GB RAM
- Apple M1
- Intel(R) Core(TM) i7-9750U CPU @ 2.60GHz 2.59 GH

Project Description

Understanding the concept of making a decision in acquiring an automobile is essential to everyone, especially first-time buyers or anyone who is inexperienced in how the automobile industry works. Usually, we need a car as a means of transportation but as we add fun to it we forget the factors that we should not underestimate, which could lead us to abandon our source of transportation and again go back to commuting, which is not a bad way but adds too many hassles and is less comfortable as if you have your own car that you could use right when you need it. Classifying a good car from a decent to a bad one is usually done manually with the help of our friendly mechanics who tells us to buy this because of this or from the opinion of our family and friends who had previously experienced car troubles. [1]

This project focuses on the top Machine Learning Models that can be applied to the Car Model Evaluation dataset in order to get the best classification results. The classifiers used are – Logistic Regression, K-nearest Neighbor, Random Forest Classifier, and Decision Tree.

Dataset used in the project –

Car Evaluation Dataset, UCI Machine Learning Repository.

Type of Data Set –

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, and safety.

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

Number of Instances: 1728

Read Dataset

The given dataset is in the format .data, we need to firstly convert it into .csv file. For that upload the car.data file to MS Excel, and save it as car.csv file.

```
import pandas as pd

df1 = pd.read_csv(r"C:\Users\prakh\Downloads\Summer Internship Project A (1)\car.csv", header = None)

df1
```

	0
0	vhigh,vhigh,2,2,small,low,unacc
1	vhigh,vhigh,2,2,small,med,unacc
2	vhigh,vhigh,2,2,small,high,unacc
3	vhigh,vhigh,2,2,med,low,unacc
4	vhigh,vhigh,2,2,med,med,unacc

Note - we need to split column by delimiter into multiple columns

```
df1.to_csv("car2.csv", header=['temp'], index = False)

df = pd.read_csv('car2.csv')

df
```

	temp
0	vhigh,vhigh,2,2,small,low,unacc
1	vhigh,vhigh,2,2,small,med,unacc
2	vhigh,vhigh,2,2,small,high,unacc
3	vhigh,vhigh,2,2,med,low,unacc
4	vhigh,vhigh,2,2,med,med,unacc
...	...
1723	low,low,5more,more,med,med,good
1724	low,low,5more,more,med,high,vgood
1725	low,low,5more,more,big,low,unacc
1726	low,low,5more,more,big,med,good
1727	low,low,5more,more,big,high,vgood

```
df[['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']] = df['temp'].str.split(',', expand=True)
```

```
df
```

	temp	buying	maint	doors	persons	lug_boot	safety	class
0	vhigh,vhigh,2,2,small,low,unacc	vhigh	vhigh	2	2	small	low	unacc
1	vhigh,vhigh,2,2,small,med,unacc	vhigh	vhigh	2	2	small	med	unacc
2	vhigh,vhigh,2,2,small,high,unacc	vhigh	vhigh	2	2	small	high	unacc
3	vhigh,vhigh,2,2,med,low,unacc	vhigh	vhigh	2	2	med	low	unacc
4	vhigh,vhigh,2,2,med,med,unacc	vhigh	vhigh	2	2	med	med	unacc
...	...	...	...	...	...	...	...	...
1723	low,low,5more,more,med,med,good	low	low	5more	more	med	med	good
1724	low,low,5more,more,med,high,vgood	low	low	5more	more	med	high	vgood
1725	low,low,5more,more,big,low,unacc	low	low	5more	more	big	low	unacc
1726	low,low,5more,more,big,med,good	low	low	5more	more	big	med	good
1727	low,low,5more,more,big,high,vgood	low	low	5more	more	big	high	vgood

```
# Drop first column of dataframe
df = df.iloc[: , 1:]

df
```

	buying	maint	doors	persons	lug_boot	safety	class
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc
...	...	...	...	...	...	...	...
1723	low	low	5more	more	med	med	good
1724	low	low	5more	more	med	high	vgood
1725	low	low	5more	more	big	low	unacc
1726	low	low	5more	more	big	med	good
1727	low	low	5more	more	big	high	vgood

Check for Missing Data

Check the info of the dataset and missing values in the dataset

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   buying    1728 non-null   object
 1   maint     1728 non-null   object
 2   doors     1728 non-null   object
 3   persons   1728 non-null   object
 4   lug_boot  1728 non-null   object
 5   safety    1728 non-null   object
 6   class     1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

Note - there are no null values in the dataset.

Exploratory Data Analysis

```
df.head(10)
```

	buying	maint	doors	persons	lug_boot	safety	class
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc
5	vhigh	vhigh	2	2	med	high	unacc
6	vhigh	vhigh	2	2	big	low	unacc
7	vhigh	vhigh	2	2	big	med	unacc
8	vhigh	vhigh	2	2	big	high	unacc
9	vhigh	vhigh	2	4	small	low	unacc

Check how the data in each category is distributed

```
for val in df.columns:
    print(df[val].value_counts())
    print("\n")
```

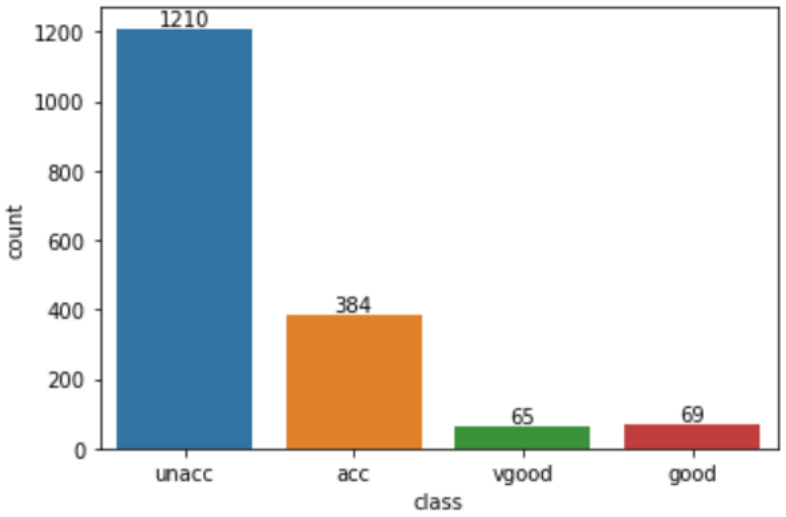
```
vhigh    432
high     432
med      432
low      432
Name: buying, dtype: int64


vhigh    432
high     432
med      432
low      432
Name: maint, dtype: int64


2       576
4       576
more    576
Name: persons, dtype: int64


small    576
med      576
big      576
Name: lug_boot, dtype: int64


low     576
med     576
high    576
Name: safety, dtype: int64


unacc    1210
acc       384
good       69
vgood      65
Name: class, dtype: int64
```

Note - The data in each column is distributed equally except for column - "Class"

```
import seaborn as sns

cp = sns.countplot(df['class'])
cp.bar_label(cp.containers[0])
```

```
C:\Users\prakh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
rg: x. From version 0.12, the only valid positional argument will be `data`,
yword will result in an error or misinterpretation.
  warnings.warn(
```

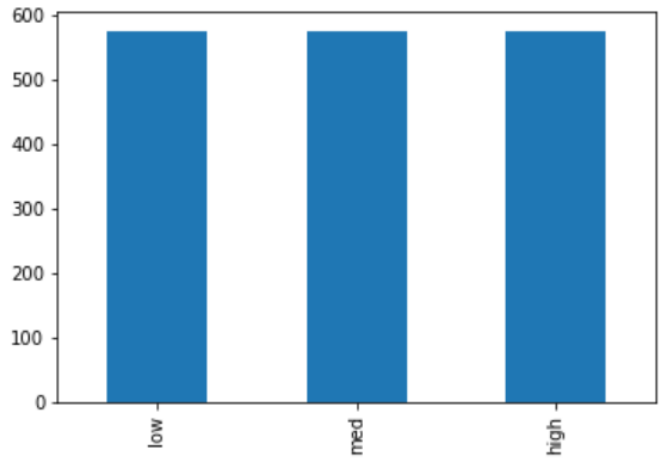
```
[Text(0, 0, '1210'), Text(0, 0, '384'), Text(0, 0, '65'), Text(0, 0, '69')]
```



The column "class" is distributed as :-

1210 - accepable cars
384 - unacceptable cars
65 - very good condition
69 - good condition

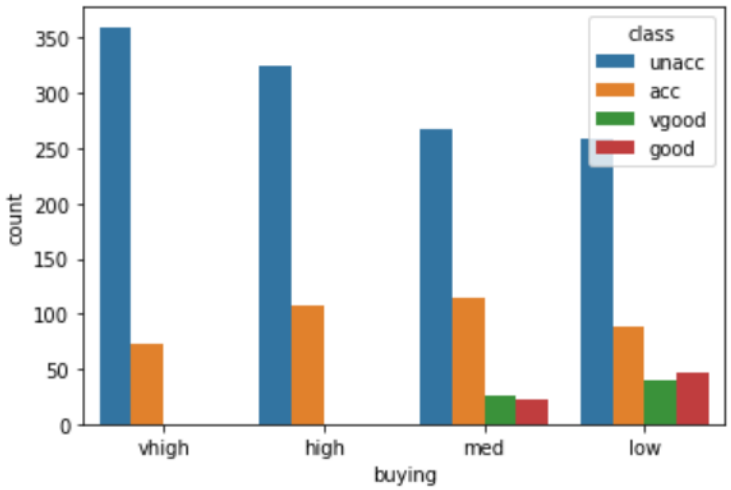
```
import matplotlib.pyplot as plt

df['safety'].value_counts().plot(kind = 'bar')
plt.show()
```



The dataset contains approximately similar number of low, medium, and high safety condition cars.

```
sns.countplot(df['buying'], hue = df['class'])
plt.show()
```

```
C:\Users\prakh\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarn
rg: x. From version 0.12, the only valid positional argument will be `data`, and
yword will result in an error or misinterpretation.
  warnings.warn(
```



We find that cars having the highest buying price has the most car unacceptability ratio, whereas medium price segment cars have the highest car acceptability ratio.

Handle Categorical Values

We know that machine learning algorithms do not work well with categorical values, so we need handle these categorical values, for that we have many encoding techniques, here we'll use labelEncoder

```
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()
```

```
for i in df.columns:
    df[i]=le.fit_transform(df[i])
```

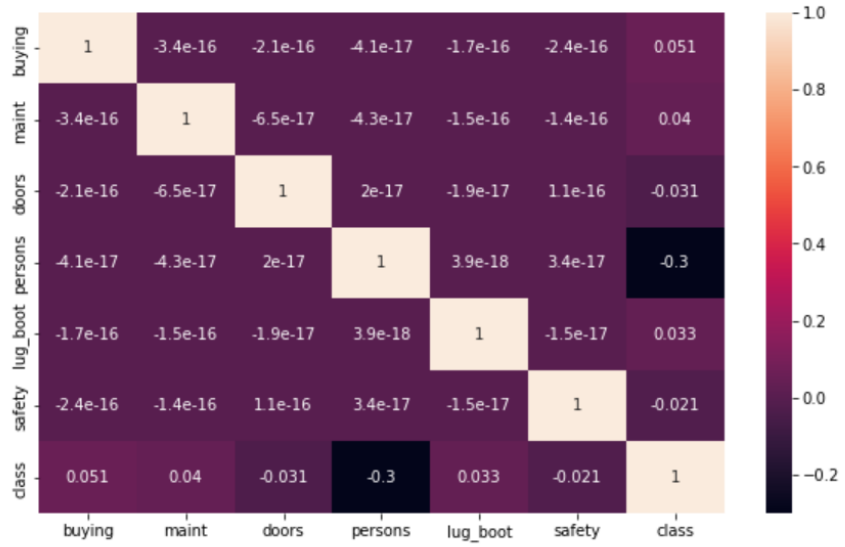
```
df
```

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

Heatmap of the columns on dataset with each other. It shows Pearson's correlation coefficient of column w.r.t other columns.

```
fig=plt.figure(figsize=(10,6))
sns.heatmap(df.corr(),annot=True)
```

```
<AxesSubplot:>
```



Pearson's Correlation Coefficient is a linear correlation coefficient that returns a value of between -1 and +1.
-1 means there is a strong negative correlation and
+1 means that there is a strong positive correlation.
0 means that there is no correlation (this is also called zero correlation).

Here analysing the heatmap we find that almost all the columns except 'person' has have zero or very weak correlation with the target class.
'person' is the only predictor variable that is correlated with the target variable 'class'

Split Data for Training and Testing

```
X=df[df.columns[:-1]]   # X contains the predictor variable
y=df['class']           # y contains the target variable
```

```
X
```

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

Splitting the dataset into training (70 %) and testing data (30 %)

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Apply different Classification Algorithms and tune them

1. Logistic Regression

```
from sklearn.linear_model import LogisticRegression
```

```
logreg=LogisticRegression(solver='newton-cg',multi_class='multinomial')

#Modified version of logistic regression that
#predicts a multinomial probability (i.e. more than two classes) for each input
```

```
logreg.fit(X_train,y_train)
```

```
LogisticRegression
LogisticRegression(multi_class='multinomial', solver='newton-cg')
```

```
lr=logreg.predict(X_test)
```

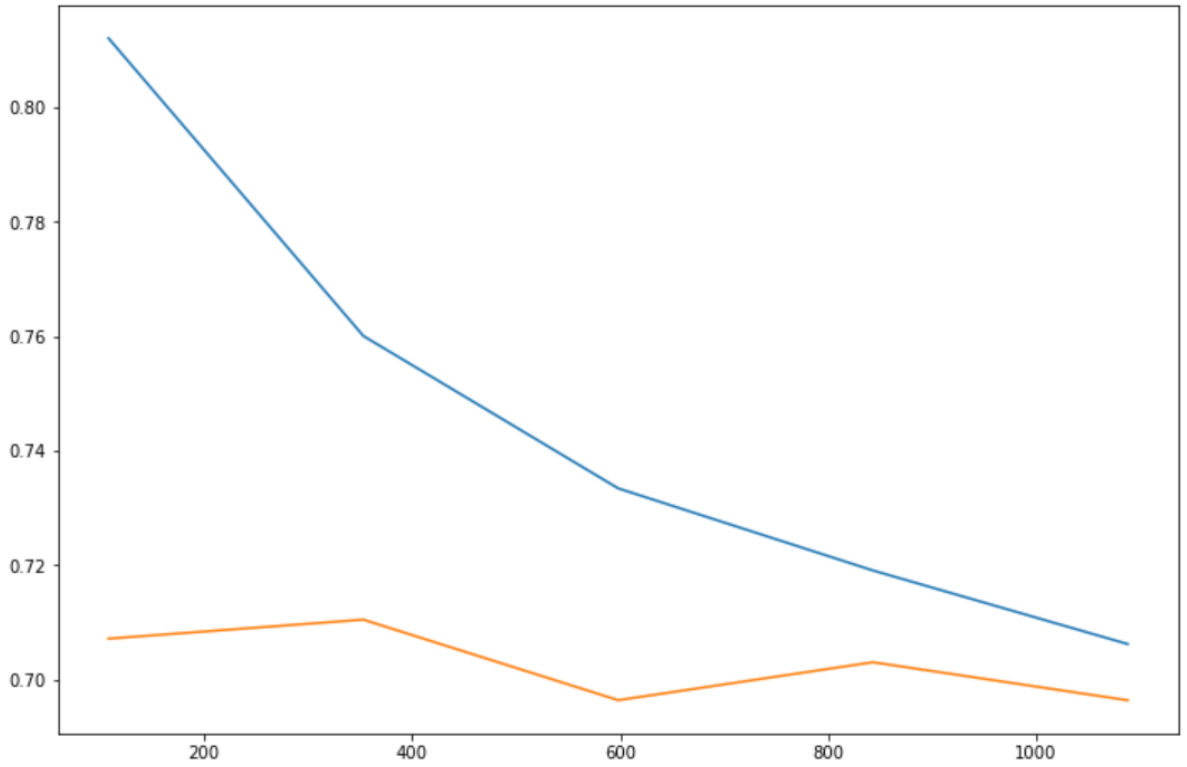
```
logreg.score(X_test,y_test)
```

```
0.6647398843930635
```

Note - Logistic Regression is giving an accuracy of 66% only.

```
#Learning curve for logistic regression model

from sklearn.model_selection import learning_curve

lc=learning_curve(logreg,X_train,y_train,cv=10,n_jobs=-1)
# cross-validation is set to 10
size=lc[0]
train_score=[lc[1][i].mean() for i in range (0,5)]
test_score=[lc[2][i].mean() for i in range (0,5)]
fig=plt.figure(figsize=(12,8))
plt.plot(size,train_score)
plt.plot(size,test_score)
```

```
[<matplotlib.lines.Line2D at 0x2e6edf21820>]
```



We find that with increasing number of smaples, training accuracy is decreasing

The reason could be severe class imbalance.

One approach to addressing imbalanced datasets is to oversample the minority class.
The simplest approach involves duplicating examples in the minority class

```
import imblearn
print(imblearn.__version__)
```

```
0.9.1
```

```
from imblearn.over_sampling import SMOTE

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 10)

sm = SMOTE(random_state = 2)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train.ravel())

clf = LogisticRegression(solver='newton-cg',multi_class='multinomial')
model_res = clf.fit(X_train_res, y_train_res)
```

```
lr2=clf.predict(X_test)
```

```
logreg.score(X_test,y_test)
```

```
0.708092485549133
```

NOTE - The accuracy has been increased from 66% to 71%.

2. k-Nearest Neighbour

```
from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier(n_jobs=-1)
```

```
knn.fit(X_train,y_train)
kn=knn.predict(X_test)
knn.score(X_test,y_test)
```

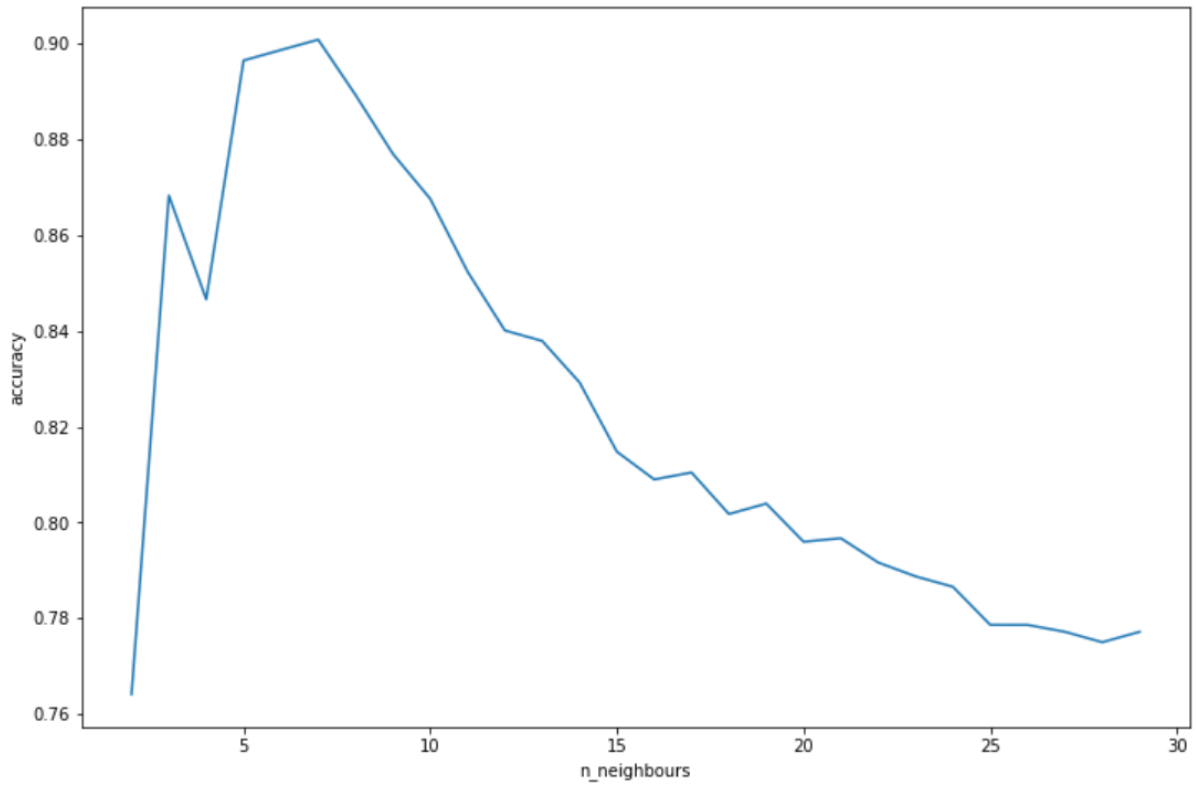
```
0.9248554913294798
```

KNN model gives us the accuracy of 92 %

```
from sklearn.model_selection import cross_val_score

avg_score=[]
for k in range(2,30):
    knn=KNeighborsClassifier(n_jobs=-1,n_neighbors=k)
    score=cross_val_score(knn,X_train,y_train,cv=5,n_jobs=-1,scoring='accuracy')
    avg_score.append(score.mean())
```

```
plt.figure(figsize=(12,8))
plt.plot(range(2,30),avg_score)
plt.xlabel("n_neighbours")
plt.ylabel("accuracy")
```

```
Text(0, 0.5, 'accuracy')
```



for n= 8, accuracy is heighest

3. Random forest classifier

```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_estimators=500)
```

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

```
RandomForestClassifier(n_estimators=500)
```

```
rfc = clf.predict(X_test)
```

```
clf.score(X_test,y_test)
```

```
0.9942196531791907
```

Random Forest Model gives accuracy of 99%

4. Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier

DT_classifier = DecisionTreeClassifier(criterion='gini',max_depth=3,min_samples_split=10)
DT_classifier.fit(X_train, y_train)
```

```
DecisionTreeClassifier(max_depth=3, min_samples_split=10)
```

```
dtc = DT_classifier.predict(X_test)
```

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test, dtc)
```

```
array([[ 59,   0,   5,   0],
       [ 13,   0,   0,   0],
       [ 37,   0, 214,   0],
       [ 18,   0,   0,   0]], dtype=int64)
```

```
DT_classifier.score(X_test,y_test)
```

```
0.7890173410404624
```

Get performance metrics for all the applied classifiers

Logistic Regression

```
print(classification_report(y_test,lr))
```

```
              precision    recall  f1-score   support

           0       0.30      0.19      0.23        64
           1       0.00      0.00      0.00        13
           2       0.76      0.92      0.83       251
           3       0.60      0.17      0.26        18

    accuracy                           0.71       346
   macro avg       0.42      0.32      0.33       346
weighted avg       0.64      0.71      0.66       346
```

f1 - score of 71%

K-nearest Neighbour

```
from sklearn.metrics import classification_report, confusion_matrix

print(classification_report(y_test,kn))
```

```
              precision    recall  f1-score   support

           0       0.91      0.77      0.83        64
           1       0.71      0.77      0.74        13
           2       0.94      1.00      0.97       251
           3       1.00      0.61      0.76        18

    accuracy                           0.92       346
   macro avg       0.89      0.79      0.82       346
weighted avg       0.93      0.92      0.92       346
```

Average f1-score is 92 %

Average f1-score is 92 %

Random Forest Classifier

```
print(classification_report(y_test,rfc))
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        64
           1       0.93      1.00      0.96        13
           2       1.00      1.00      1.00       251
           3       1.00      0.94      0.97        18

    accuracy                           0.99       346
   macro avg       0.98      0.98      0.98       346
weighted avg       0.99      0.99      0.99       346
```

f1 - score of 99%

Decision Tree

```
print(metrics.classification_report(y_test, dtc))
```

```
              precision    recall  f1-score   support

           0       0.46      0.92      0.62        64
           1       0.00      0.00      0.00        13
           2       0.98      0.85      0.91       251
           3       0.00      0.00      0.00        18

    accuracy                           0.79       346
   macro avg       0.36      0.44      0.38       346
weighted avg       0.79      0.79      0.77       346
```

Note - Using Decision Tree model, we get an accuracy of 79%

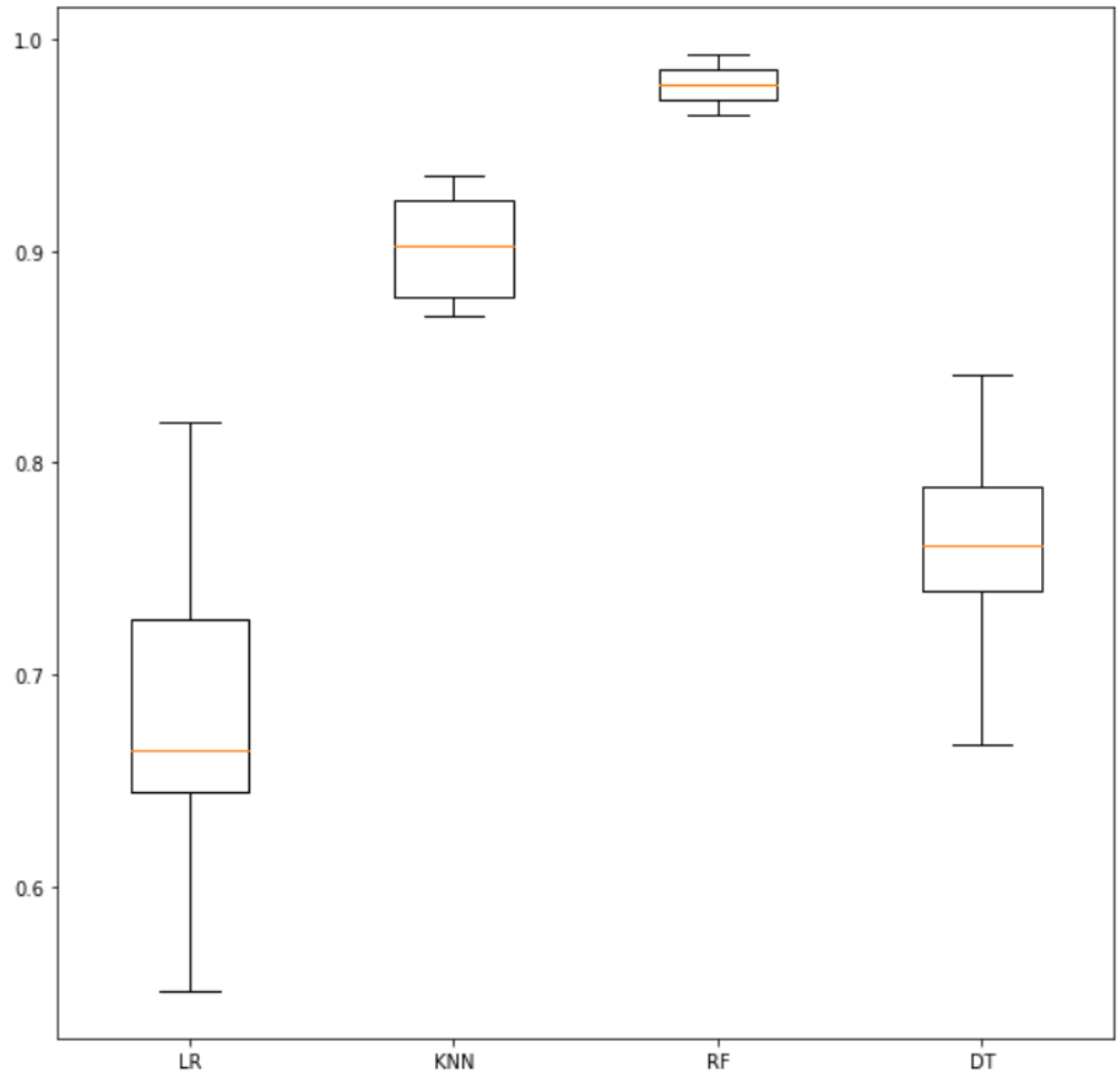
Visually compare the performance of all classifiers

```
models = []
models.append(('LR', LogisticRegression(solver='newton-cg',multi_class='multinomial')))
models.append(('KNN', KNeighborsClassifier(n_jobs=-1)))
models.append(('RF', RandomForestClassifier(n_estimators=500)))
models.append(('DT', DecisionTreeClassifier( criterion= 'gini', max_depth= 3, min_samples_split= 10)))
```

```
from sklearn import model_selection
from random import seed

results = []
names = []
scoring = 'accuracy'
for name, model in models:
    kfold = model_selection.KFold(n_splits=10)
    cv_results = model_selection.cross_val_score(model, X_train, y_train, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: 0.680138 (0.073007)
KNN: 0.902268 (0.023778)
RF: 0.977562 (0.009430)
DT: 0.757517 (0.047722)
```

```
fig = plt.figure(figsize=(10,10))
fig.suptitle('compare sklearn classification algorithms')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```



NOTE - RANDOM FOREST is the best machine learning model in this case with an accuracy of 99%

List of References

[1] - Fatalla, Ronald. (2004). Decision Model for Car Evaluation Final Project in Pattern Recognition.

[2] – Vogt, Mike. “How to compare sklearn classification algorithms in Python?”. 31/05/2022. https://www.projectpro.io/recipes/compare-sklearn-classification-algorithms-in-python

[3] – Sushant, Sonar. “Car-Evaluation-Dataset-Classification”. 15/11/2017. https:// github.com/sonarsushant/Car-Evaluation-Dataset-Classification.git

[4] – Masum, Mohammed. “Car Evaluation Analysis Using Decision Tree Classifier”. 12/02/2022. https://towardsdatascience.com/car-evaluation-analysis-using-decision-tree-classifier-61a8ff12bf6f

[5] – Salazar, Roberto. “Machine Learning Classifiers Comparison with Python”. 04/06/2020. https://towardsdatascience.com/machine-learning-classifiers-comparison-with-python-331499aecdbca