Abstract

With the dataset used in this report, we will try to predict the chances of a student getting an admission based on different exam scores and other descriptive features. We have used three classification algorithms - K nearest neighbours, Gaussian naive bayes and Decision tree algorithms to test our training dataset which are split into 80:20 to test all three algorithms.

Introduction

The aim of this project is to predict if an individual has any chances of getting admitted in given universities based on the relationship of descriptive features with our target variable(chance of admission). In this report, we will focus on classification problem to predict our target variable. We have done the data pre-processing, data visualisation in the phase 1 of our project.

Methodology

The objective of this project is to anticipate if a student will get admitted to a university or not depending on the available descriptive features.

We create predicitve models to achieve the necessary outcome .The source of data is from kaggle at https://www.kaggle.com/mohansacharya/graduate-admissions)(Kaggle.com (https://www.kaggle.com/mohansacharya/graduate-admissions)(Kaggle.com), 2019).The descriptive features include 6 numeric and 1 categorical feature. The target feature has 2 classes 1(Will Get Admitted) or 0(Wont get admitted). The total dataset consist of 500 observations.

In this study we will be using the following classifier for target prediction. K-Nearest Neighbors (KNN), Naive Bayes (NB) Decision trees (DT) This phase begins with data scaling and model fitting. The data cleaning and exploration was done in the Phase 1 of the project. All the descriptive features have been Normalized. The data is split as 80-20. i.e 80 for training and 20 for test.Hence our training data has 400 observations while the test data set has 100 observations.

We have done the feature selection using the Random forest importance method.

Dataset Features

In [ ]:

```
Following are the descriptive features from Admission_Predict_Ver1.1 file:

1) Serial No. (1 to 500).

2) GRE Score (out of 340).

3) TOEFL Score (out of 120).

4) University Rating (out of 5).

5) SOP (out of 5): Statement of Purpose.

6) LOR (out of 5): Letter of Recomendation.

7) CGPA (out of 10).

8) Research (either 0 or 1): 0 means student has no experience in research while vice v
ersa in 1
```

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings("ignore")
import io
import requests
import os, ssl
```

In [2]:

```
Admission_file = 'Admission_Predict_Ver1.1.csv'
```

In [3]:

```
import os
os.getcwd()
os.chdir('C:\Users\User\Desktop\Masters\Sem2\Machine Learning\Project Phase 2')
```

In [4]:

```
Admissions = pd.read_csv(Admission_file,decimal='.',skipinitialspace=True)
```

In [5]:

```
Admissions.head()
```

Out[5]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

In [6]:

```
Admissions.columns=Admissions.columns.str.strip()
```

In [7]:

```
Admissions.columns.values
```

Out[7]:

```
array(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating',
       'SOP', 'LOR', 'CGPA', 'Research', 'Chance of Admit'], dtype=object)
```

In [8]:

```
Admissions=Admissions.drop(['Serial No.'],axis=1)
```

In [9]:

```
Admissions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
GRE Score            500 non-null int64
TOEFL Score          500 non-null int64
University Rating    500 non-null int64
SOP                  500 non-null float64
LOR                  500 non-null float64
CGPA                 500 non-null float64
Research             500 non-null int64
Chance of Admit      500 non-null float64
dtypes: float64(4), int64(4)
memory usage: 31.3 KB
```

In [10]:

```
Admissions.describe()
```

Out[10]:

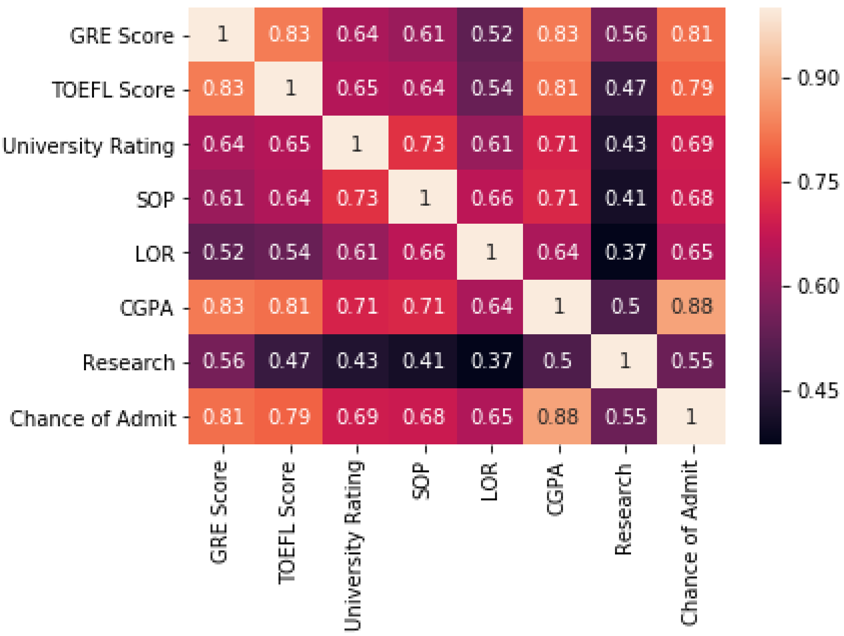
	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	
count	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	5
mean	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	
std	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	
min	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	
25%	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	
50%	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	
75%	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	
max	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	

In [11]:

```
sns.heatmap(Admissions.corr(),annot=True)
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xb7e6b00>
```



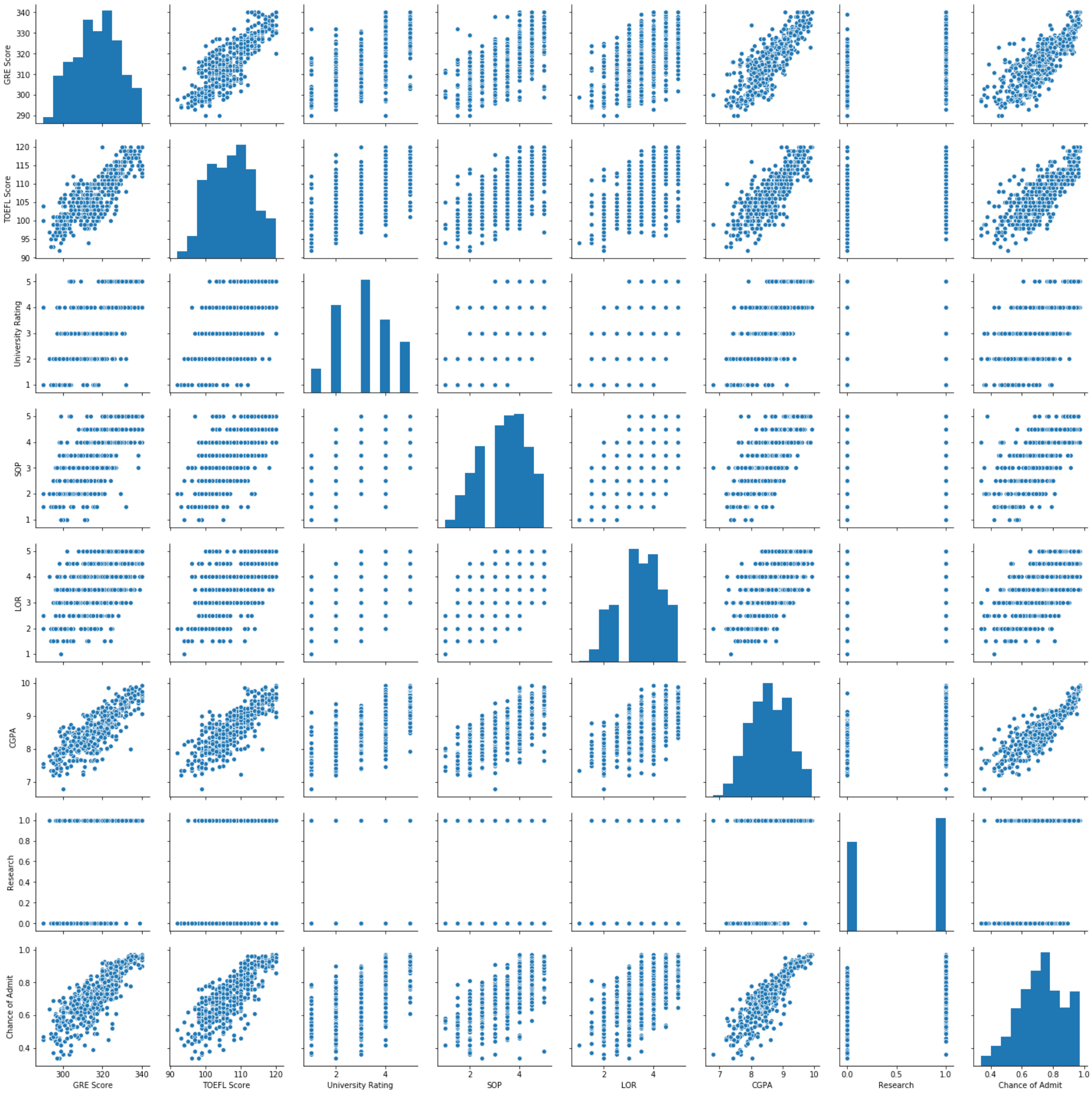
Looking at the heatmap we could see that the attributes CGPA, GRE and TOEFL Score has the strongest correlation with the target feature.

In [12]:

```
sns.pairplot(Admissions)
```

Out[12]:

```
<seaborn.axisgrid.PairGrid at 0xc145550>
```



Looking at the scatterplot we could see that the attributes CGPA, GRE and TOEFL Score has a positive linear relationship with the target feature.

Normalization

In [13]:

```
Admissions_desc = Admissions.drop(["Chance of Admit"],axis=1)

target = Admissions["Chance of Admit"]

target = [1 if each > 0.8 else 0 for each in target]
```

In [ ]:

```
from sklearn import preprocessing

Admissions_desc_orig = Admissions_desc.copy()

scaler = preprocessing.MinMaxScaler()
scaler.fit(Admissions_desc)
Admissions = scaler.fit_transform(Admissions_desc)
```

In [15]:

```
from sklearn.ensemble import RandomForestClassifier

no_features = 7
model_rfc = RandomForestClassifier(n_estimators=100)
model_rfc.fit(Admissions_desc,target)
fs_indices_rfc = np.argsort(model_rfc.feature_importances_)[::-1][0:no_features]

best_feat_rfc = Admissions_desc_orig.columns[fs_indices_rfc].values
best_feat_rfc
```

Out[15]:

```
array(['CGPA', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR', 'Research'], dtype=object)
```

In [16]:

```
feature_importances_rfc = model_rfc.feature_importances_[fs_indices_rfc]
feature_importances_rfc
```

Out[16]:

```
array([0.37931279, 0.17623953, 0.17423641, 0.10993025, 0.09062522,
       0.04390624, 0.02574956])
```

Based on the outcome of Heatmap, Scatterplot and Random Classifier we conclude that CGPA, TOEFL and GRE scores are the most relevant features to predict the target variable.

In [ ]:

```
X = Admissions_desc[['GRE Score', 'TOEFL Score','CGPA']]
```

Creating training and test data

In [19]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,target,test_size = 0.20,random_st
ate = 42)
```

For 80% Train and 20% Test

Finding the k value:

As a result of the test, the best k value is 3.

Confusion Matrix:

For Actual 1: 27

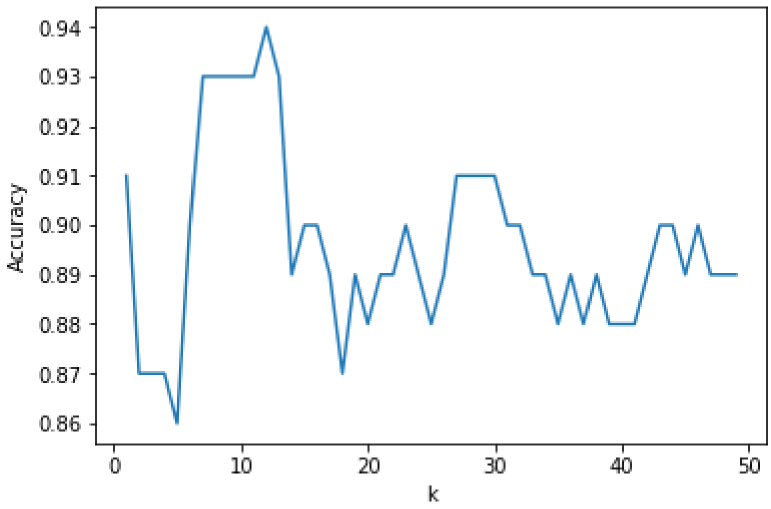
```
Predicted 1: 21
Predicted 0: 6
```

For Actual 0: 73

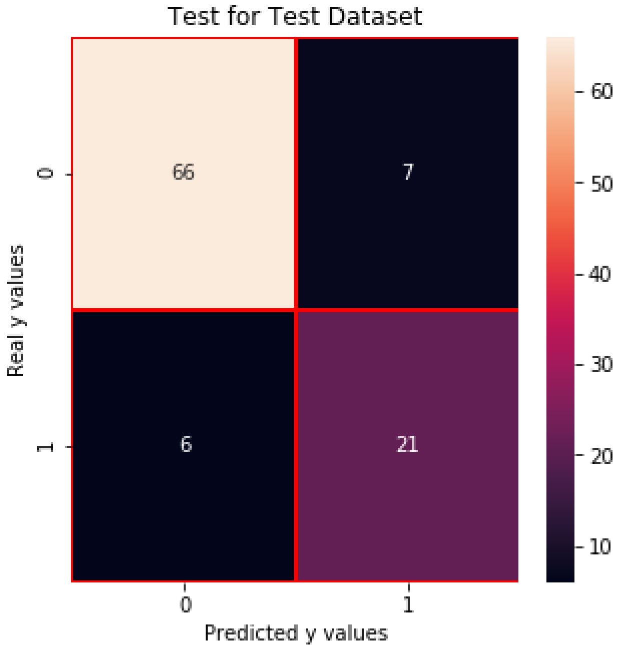
```
Predicted 1: 7
Predicted 0: 66
```

In [21]:

```
from sklearn.neighbors import KNeighborsClassifier

# finding k value
scores = []
for each in range(1,50):
    knn_n = KNeighborsClassifier(n_neighbors = each)
    knn_n.fit(X_train,y_train)
    scores.append(knn_n.score(X_test,y_test))

plt.plot(range(1,50),scores)
plt.xlabel("k")
plt.ylabel("Accuracy")
plt.show()

knn = KNeighborsClassifier(n_neighbors = 3) # n_neighbors = k
knn.fit(X_train,y_train)
print("Score of 3:",knn.score(X_test,y_test))
print("Real value of y_test_01[1]: " + str(y_test[1]) + " -> The predict: " + str(knn.p
redict(X_test.iloc[[1],:])))
print("Real value of y_test_01[2]: " + str(y_test[2]) + " -> The predict: " + str(knn.p
redict(X_test.iloc[[2],:])))

# confusion matrix
from sklearn.metrics import confusion_matrix
cm_knn = confusion_matrix(y_test,knn.predict(X_test))
# print("y_test_01 == 1 :" + str(len(y_test_01[y_test_01==1]))) # 29

# cm visualization
import seaborn as sns
import matplotlib.pyplot as plt
f, ax = plt.subplots(figsize =(5,5))
sns.heatmap(cm_knn,annot = True,linewidths=0.5,linecolor="red",fmt = ".0f",ax=ax)
plt.title("Test for Test Dataset")
plt.xlabel("Predicted y values")
plt.ylabel("Real y values")
plt.show()

from sklearn.metrics import precision_score, recall_score
print("Precision: ", precision_score(y_test,knn.predict(X_test)))
print("Recall: ", recall_score(y_test,knn.predict(X_test)))

from sklearn.metrics import f1_score
print("F1-Score: ",f1_score(y_test,knn.predict(X_test)))
```



```
('Score of 3:', 0.87)
Real value of y_test_01[1]: 1 -> The predict: [0]
Real value of y_test_01[2]: 0 -> The predict: [0]
```



```
('Precision: ', 0.75)
('Recall: ', 0.7777777777777778)
('F1-Score: ', 0.7636363636363638)
```

Gaussian NB 80/20

Confusion Matrix:

For Actual 1: 27

```
Predicted 1: 26
Predicted 0: 1
```

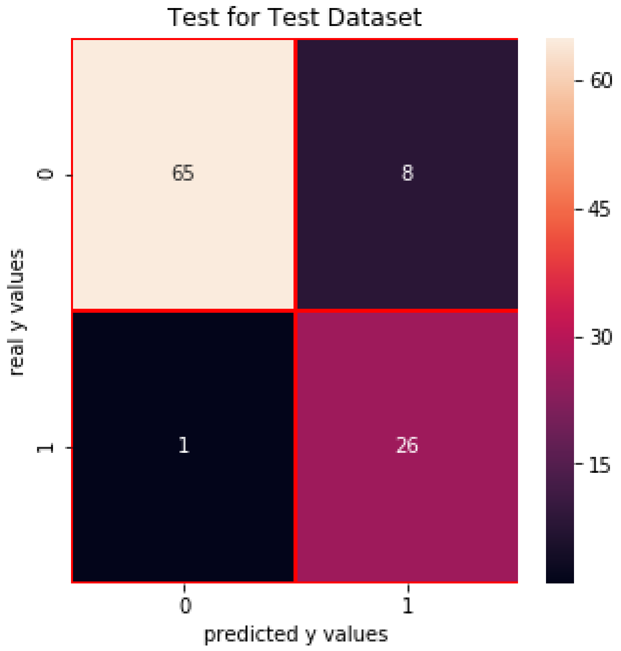
For Actual 0: 73

```
Predicted 1: 8
Predicted 0: 65
```

In [23]:

```
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train,y_train)
print("score: ", nb.score(X_test,y_test))
print("real value of y_test[1]: " + str(y_test[1]) + " -> the predict: " + str(nb.predict(X_test.iloc[[1],:])))
print("real value of y_test[2]: " + str(y_test[2]) + " -> the predict: " + str(nb.predict(X_test.iloc[[2],:])))

# confusion matrix
from sklearn.metrics import confusion_matrix
cm_nb = confusion_matrix(y_test,nb.predict(X_test))
# print("y_test == 1 :" + str(len(y_test[y_test==1]))) # 29
# cm visualization
import seaborn as sns
import matplotlib.pyplot as plt
f, ax = plt.subplots(figsize =(5,5))
sns.heatmap(cm_nb,annot = True,linewidths=0.5,linecolor="red",fmt = ".0f",ax=ax)
plt.title("Test for Test Dataset")
plt.xlabel("predicted y values")
plt.ylabel("real y values")
plt.show()

from sklearn.metrics import precision_score, recall_score
print("precision_score: ", precision_score(y_test,nb.predict(X_test)))
print("recall_score: ", recall_score(y_test,nb.predict(X_test)))

from sklearn.metrics import f1_score
print("f1_score: ",f1_score(y_test,nb.predict(X_test)))
```

```
('score: ', 0.91)
real value of y_test[1]: 1 -> the predict: [0]
real value of y_test[2]: 0 -> the predict: [0]
```



```
('precision_score: ', 0.7647058823529411)
('recall_score: ', 0.9629629629629629)
('f1_score: ', 0.8524590163934426)
```

Decision Tree Classifier

For 80% Train and 20% Test

Confusion Matrix:

For Actual 1: 27

```
Predicted 1: 24
Predicted 0: 3
```

For Actual 0: 73

```
Predicted 1: 6
Predicted 0: 67
```

In [25]:

```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
fit = dtc.fit(X_train, y_train)
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(dtc.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(dtc.score(X_test, y_test)))
```

```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.91
```

In [26]:

```
y_pre = fit.predict(X_test)
y_pre
```

Out[26]:

```
array([1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0])
```

In [27]:

```
cm = confusion_matrix(y_test,y_pre)
print cm
```

```
[[67  6]
 [ 3 24]]
```

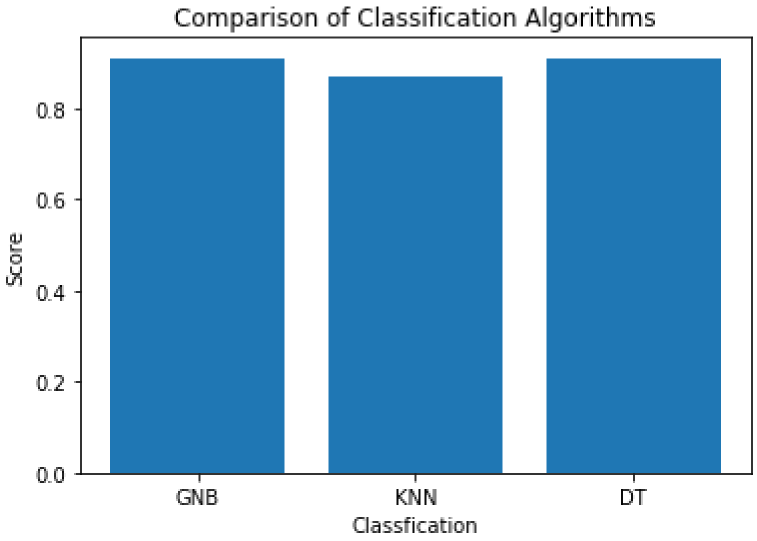
In [28]:

```
from sklearn.metrics import classification_report
print classification_report(y_test, y_pre)
```

```
              precision    recall  f1-score   support

           0       0.96      0.92      0.94        73
           1       0.80      0.89      0.84        27

   micro avg       0.91      0.91      0.91       100
   macro avg       0.88      0.90      0.89       100
weighted avg       0.91      0.91      0.91       100
```

In [29]:

```
y = np.array([nb.score(X_test,y_test),knn.score(X_test,y_test),dtc.score(X_test,y_test
)])
#x = ["GaussianNB","KNeighborsClassifier","Decision Tree"]
x = ["GNB","KNN","DT"]

plt.bar(x,y)
plt.title("Comparison of Classification Algorithms")
plt.xlabel("Classfication")
plt.ylabel("Score")
plt.show()
```



Conclusion

The aim of this assignment was to predict if a student is able to get admitted into a university based on theoir CGPA, GRE and TOEFL scores. We proceeded to model the data set by running it through K Nearest neighbours, Gaussian Bayes anjd decision tree algorithms. We achieved the best result i.e. 91% accuracy on a 80-20 train-test split on GNB algorithm.

Citation

Mohan S Acharya, Asfia Armaan, Aneeta S Antony : A Comparison of Regression Models for Prediction of Graduate Admissions, IEEE International Conference on Computational Intelligence in Data Science 2019