Krish Sukhani

Batch D , 59

DWM EXP5

```
import numpy as np
import pandas as pd
```

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # data visualization
import seaborn as sns # statistical data visualization
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier
```

```
from google.colab import drive
drive.mount("/content/gdrive")
```

```
    Mounted at /content/gdrive
```

```
df = pd.read_csv('/content/gdrive/My Drive/datasets/weatherAUS.csv',encoding= 'unicode_escape')
```

Data Cleaning

```
categorical = [var for var in df.columns if df[var].dtype=='O']
print('There are {} categorical variables\n'.format(len(categorical)))
print('The categorical variables are :', categorical)
```

```
    There are 7 categorical variables

    The categorical variables are : ['Date', 'Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm', 'RainToday
```

```
cat1 = [var for var in categorical if df[var].isnull().sum()!=0]
print(df[cat1].isnull().sum())
```

```
    WindGustDir     9330
    WindDir9am     10013
    WindDir3pm      3778
    RainToday       1406
    dtype: int64
```

```
for var in categorical:
    print(var + ' conatins '+str(len(df[var].unique()))+ " labels ")
```

```
    Date conatins 3436 labels
    Location conatins 49 labels
    WindGustDir conatins 17 labels
    WindDir9am conatins 17 labels
    WindDir3pm conatins 17 labels
```

```
RainToday conatins 3 labels
RainTomorrow conatins 2 labels
```

Splitting the Date column into respective 'Year','Month' & 'Day'.**

```
df['Date'] = pd.to_datetime(df['Date'])
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day

df.drop('Date',axis=1,inplace=True)
```

```
categorical = [var for var in df.columns if df[var].dtype=='O']
print("There are {} categorical variables : ".format(len(categorical)))
print(categorical)
```

```
There are 6 categorical variables :
['Location', 'WindGustDir', 'WindDir9am', 'WindDir3pm', 'RainToday', 'RainTomorrow']
```

Replacing the missing categorical values by the most frequent value in respective columns.

```
for var in categorical:
    df[var].fillna(df[var].mode()[0],inplace=True)
```

```
numerical = [var for var in df.columns if df[var].dtype!='O']
print(numerical)
```

```
['MinTemp', 'MaxTemp', 'Rainfall', 'Evaporation', 'Sunshine', 'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3p
```

```
num1 = df[numerical].isnull().sum()
num1 = num1[num1!=0]
num1
```

```
MinTemp            637
MaxTemp            322
Rainfall          1406
Evaporation      60843
Sunshine         67816
WindGustSpeed     9270
WindSpeed9am      1348
WindSpeed3pm      2630
Humidity9am       1774
Humidity3pm       3610
Pressure9am      14014
Pressure3pm      13981
Cloud9am         53657
Cloud3pm         57094
Temp9am            904
Temp3pm           2726
dtype: int64
```

Replacing the missing numercial values by the mean of their respective columns.

```
for col in num1.index:
    col_mean = df[col].mean()
    df[col].fillna(col_mean,inplace=True)
```

```
le = LabelEncoder()
new_df = df
for col in categorical:
```

```
    new_df[col] = le.fit_transform(df[col])
col_names = new_df.columns
```

```
new_df.head()
```

	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	Win
0	2	13.4	22.9	0.6	5.469824	7.624853	13	44.0	13	
1	2	7.4	25.1	0.0	5.469824	7.624853	14	44.0	6	
2	2	12.9	25.7	0.0	5.469824	7.624853	15	46.0	13	
3	2	9.2	28.0	0.0	5.469824	7.624853	4	24.0	9	
4	2	17.5	32.3	1.0	5.469824	7.624853	13	41.0	1	

Feature Scaling using MinMaxScaler

```
from sklearn.preprocessing import MinMaxScaler
ss = MinMaxScaler()
new_df = ss.fit_transform(new_df)
new_df = pd.DataFrame(new_df,columns = col_names )
```

```
new_df.describe()
```

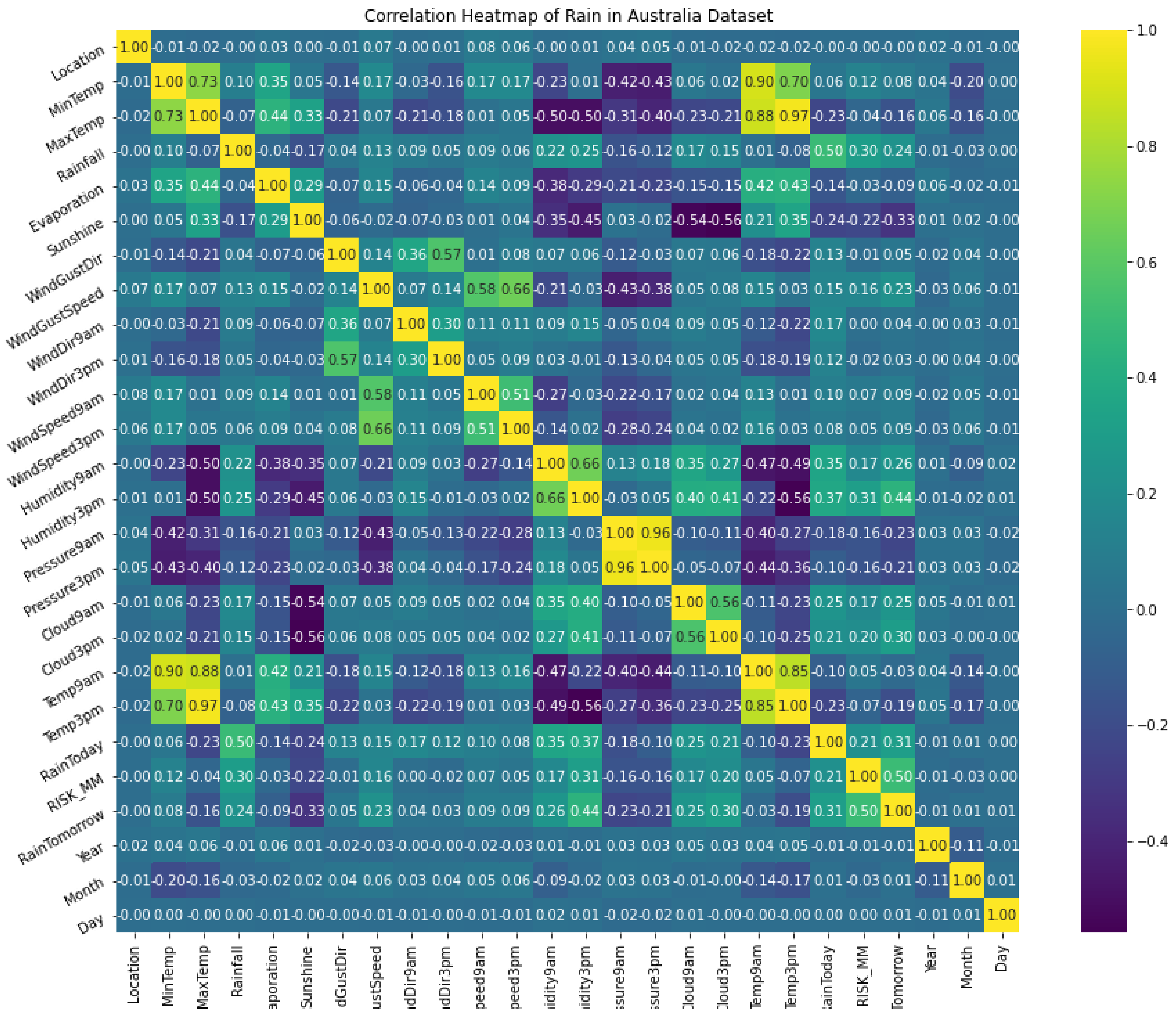
	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDi
count	142193.000000	142193.000000	142193.000000	142193.000000	142193.000000	142193.000000	142193.00000
mean	0.494597	0.487887	0.529807	0.006334	0.037723	0.525852	0.5372
std	0.296615	0.150682	0.134396	0.022704	0.021849	0.188616	0.3129
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	0.229167	0.379717	0.429112	0.000000	0.027586	0.525852	0.2666
50%	0.500000	0.483491	0.519849	0.000000	0.037723	0.525852	0.6000
75%	0.750000	0.596698	0.623819	0.002156	0.037723	0.600000	0.8666
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.0000

```
# new_df.to_csv("weatherCleaned.csv")
```

Data Visualization

Heatmap of correlation among the columns of data.

```
correlation = new_df.corr()
plt.figure(figsize=(16,12))
plt.title('Correlation Heatmap of Rain in Australia Dataset')
ax = sns.heatmap(correlation, square=True, annot=True, fmt='.2f', linecolor='white',cmap='viridis')
ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
ax.set_yticklabels(ax.get_yticklabels(), rotation=30)
plt.show()
```




```
y = new_df.RainTomorrow
X = new_df.drop('RainTomorrow',axis=1)
```

```
results = []
```

Splitting into training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=42,shuffle=True)
```

Applying various classifying algorithms on the training set and predicting the RainTomorrow using training set.

1.1 Gaussian Naive Bayes

```
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)
gnb.score(X_test,y_test)
```

```
0.9495938675762158
```

```
print(accuracy_score(y_test,y_pred))
```

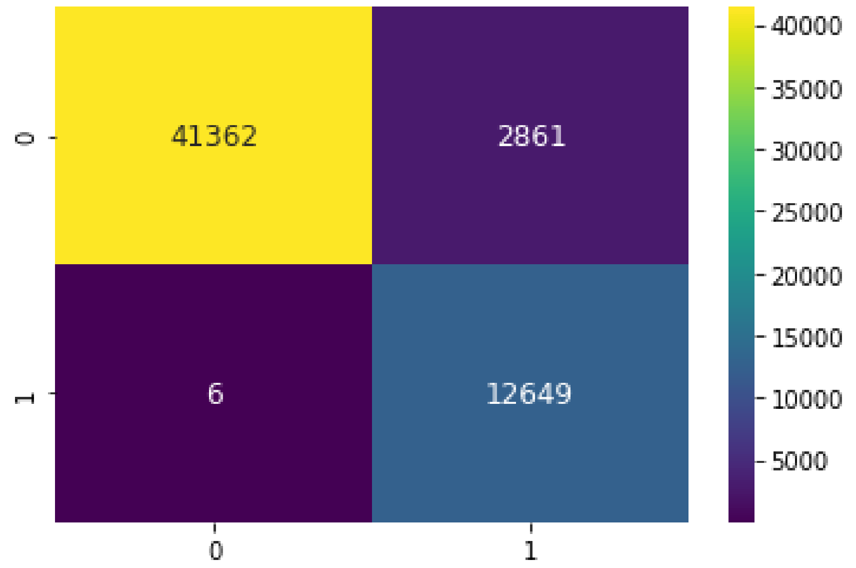
```
print(cross_val_score(gnb,X_train,y_train,cv=3))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
results.append(accuracy_score(y_test,y_pred))
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,annot_kws={"size": 12},cmap='viridis',fmt="d")
```

```
0.9495938675762158
[0.94940047 0.95277446 0.95157887]
[[41362  2861]
 [    6 12649]]
              precision    recall  f1-score   support

         0.0       1.00      0.94      0.97     44223
         1.0       0.82      1.00      0.90     12655

    accuracy                           0.95     56878
   macro avg       0.91      0.97      0.93     56878
weighted avg       0.96      0.95      0.95     56878

<matplotlib.axes._subplots.AxesSubplot at 0x7f0300b639d0>
```



Observations :

> GaussianNB implements gaussian naive bayes algorithm for classification.
>
> It assumes the maximum likelihood of the features to be Gaussian and classifies the dataset accordingly.
>
> The confusion matrix depicts that 2861 are False Positives and 6 are False Negatives.
>
> Thus, Gaussian Naive Bayes algorithm is able to predict rain tommorrow with accuracy of 94.95%.

1.2 Decision Trees

```
dtc = DecisionTreeClassifier(max_depth=10, min_samples_split=2,random_state=42)
dtc.fit(X_train,y_train)
y_pred = dtc.predict(X_test)
dtc.score(X_test,y_test)
```

```
1.0
```

```
print(accuracy_score(y_test,y_pred))
print(cross_val_score(dtc,X_train,y_train,cv=3))
print(confusion_matrix(y_test,y_pred))
```

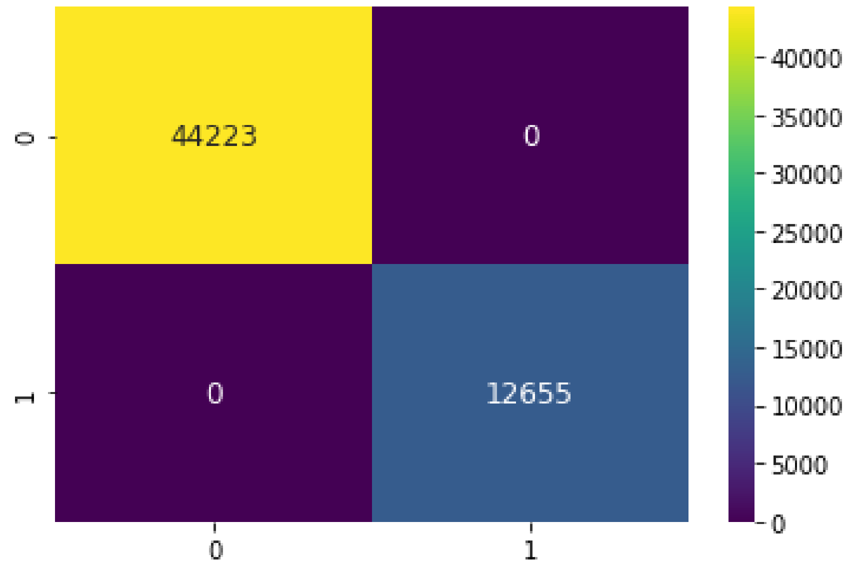
```
print(classification_report(y_test,y_pred))
results.append(accuracy_score(y_test,y_pred))
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,annot_kws={"size": 12},cmap='viridis',fmt="d")
```

```
1.0
[1. 1. 1.]
[[44223     0]
 [    0 12655]]
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     44223
         1.0       1.00      1.00      1.00     12655

    accuracy                           1.00     56878
   macro avg       1.00      1.00      1.00     56878
weighted avg       1.00      1.00      1.00     56878

<matplotlib.axes._subplots.AxesSubplot at 0x7f0300606610>
```



Observations :

> Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression.
>
> The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.
>
> DecisionTreeClassifier is capable of both binary classification and multiclass classification.
>
> The confusion matrix shows that there are 0 FP or FN.
>
> Hence, the DecisionTreeClassifier is able to predict rain tomorrow with an impressive accuracy of 100%.

1.3 Support Vector Machines

```
svc = LinearSVC(random_state=42)
svc.fit(X_train,y_train)
y_pred = svc.predict(X_test)
svc.score(X_test,y_test)
print(cross_val_score(svc,X_train,y_train,cv=3))
```

```
[0.93772636 0.93839229 0.93874393]
```

```
print(accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

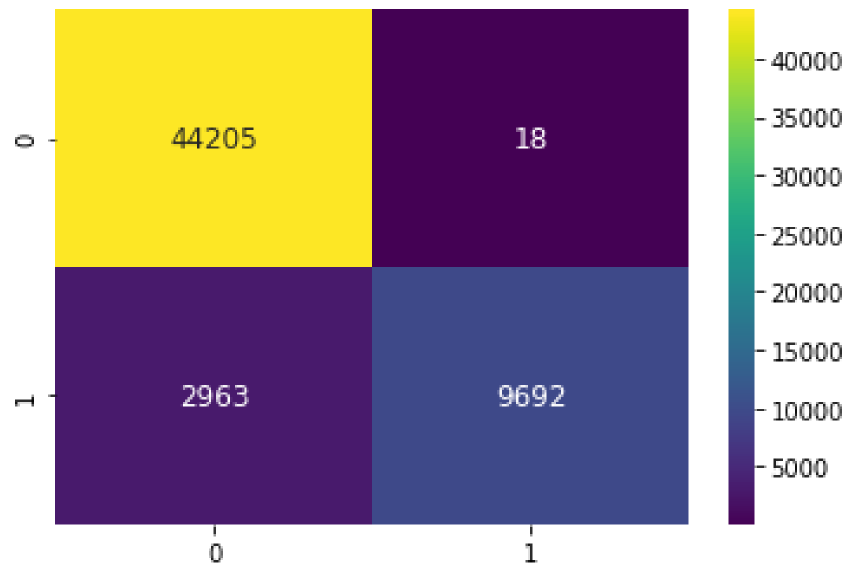
```
print(classification_report(y_test,y_pred))
results.append(accuracy_score(y_test,y_pred))
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,annot_kws={"size": 12},cmap='viridis',fmt="d")
```

```
0.9475895776926052
[[44205    18]
 [ 2963  9692]]
              precision    recall  f1-score   support

         0.0       0.94      1.00      0.97     44223
         1.0       1.00      0.77      0.87     12655

    accuracy                           0.95     56878
   macro avg       0.97      0.88      0.92     56878
weighted avg       0.95      0.95      0.94     56878

<matplotlib.axes._subplots.AxesSubplot at 0x7f02fed59c50>
```



Observations :

> > Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection.
> >
> > LinearSVC is another implementation of Support Vector Classification for the case of a linear kernel.
> >
> > This class supports both dense and sparse input and the multiclass support is handled according to a one-vs-the-rest scheme.
> >
> > From the confusion matrix it is evident that 18 are FP and 2963 are FN.
> >
> > Therby, the LinearSVC is able to predict rain tomorrow with 94.75% accuracy.

1.4 Random Forest

```
rfc = RandomForestClassifier(n_estimators=200,max_depth=10, random_state=42)
rfc.fit(X_train,y_train)
y_pred = rfc.predict(X_test)
rfc.score(X_test,y_test)
```

```
0.9999296740391715
```

```
print(accuracy_score(y_test,y_pred))
print(cross_val_score(rfc,X_train,y_train,cv=3))
print(confusion_matrix(y_test,y_pred))
```

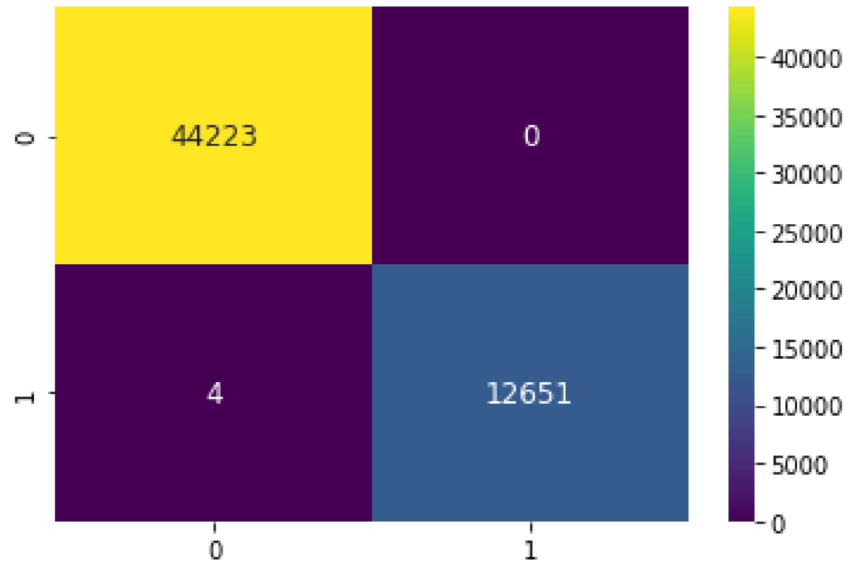
```
print(classification_report(y_test,y_pred))
results.append(accuracy_score(y_test,y_pred))
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True,annot_kws={"size": 12},cmap='viridis',fmt="d")
```

```
0.9999296740391715
[0.99996484 0.99996484 0.99996484]
[[44223     0]
 [    4 12651]]
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     44223
         1.0       1.00      1.00      1.00     12655

    accuracy                           1.00     56878
   macro avg       1.00      1.00      1.00     56878
weighted avg       1.00      1.00      1.00     56878

<matplotlib.axes._subplots.AxesSubplot at 0x7f0310714690>
```



Observations :

> In random forests (RandomForestClassifier and RandomForestRegressor classes), each tree in the ensemble is built from a sample drawn with replacement.
>
> A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.
>
> The confusion matrix depicts that there are only 4 FN.
>
> Hence, the RandomForestClassifier is able to predict rain tomorrow with 99.99% accuracy.

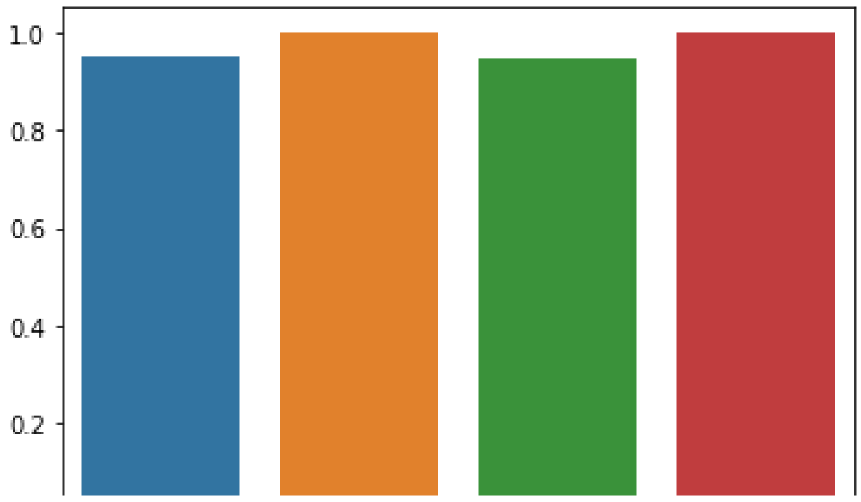
Comaprison of Various Classifying algorithms

```
names = ["Naive Bayes","Decision Tree","Linear SVM","Random Forest",]
results
```

```
[0.9495938675762158, 1.0, 0.9475895776926052, 0.9999296740391715]
```

```
sns.barplot(names,results)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variab
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f02fec35a10>
```



Conclusion :

> > The Decison Tree Algorithm outperforms other algorithms in terms of precison,accuracy and recall.
> >
> > Also,LinearSVM is the lowest in terms of accuracy.
> >
> > Gaussian Naive Bayes performs well in case of binary classification.
> >
> > Thus, Random Forest and Decision Trees are best suited for binary classification problems.