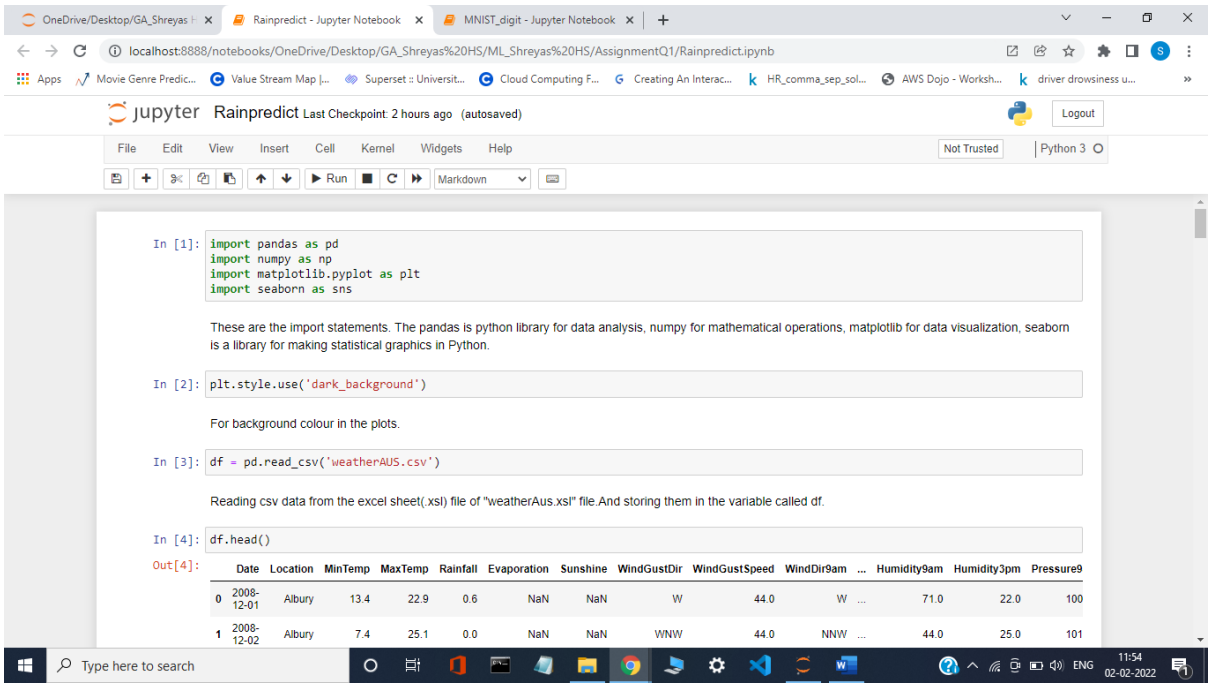
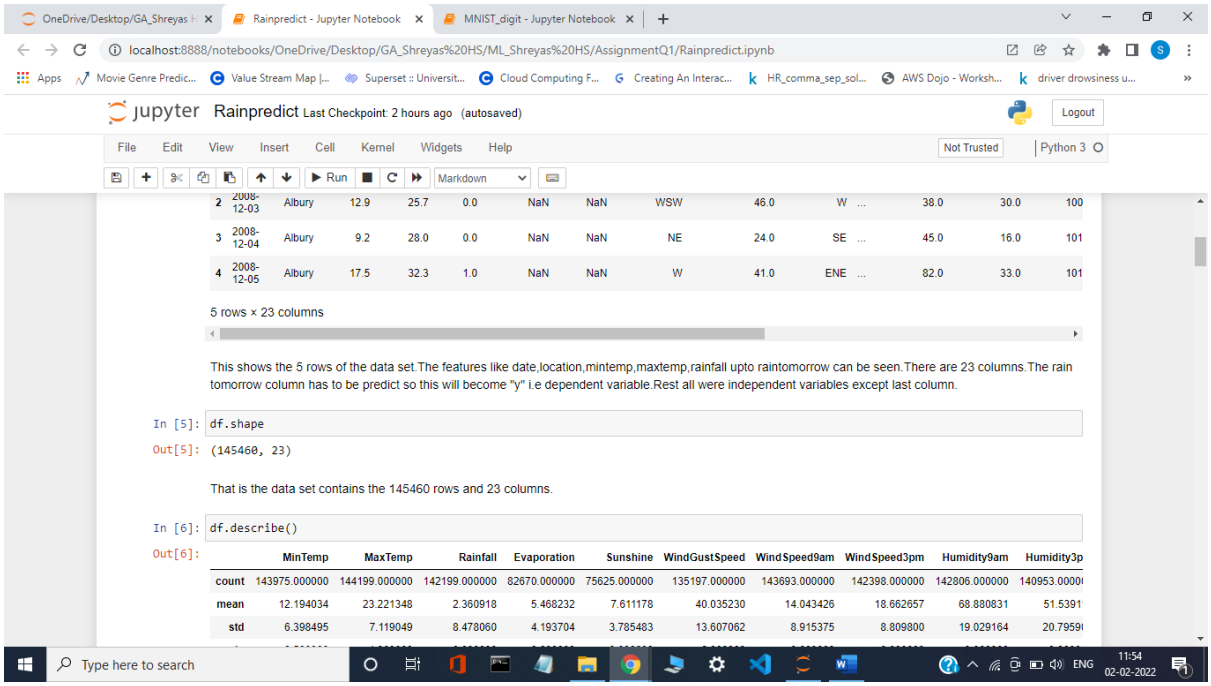
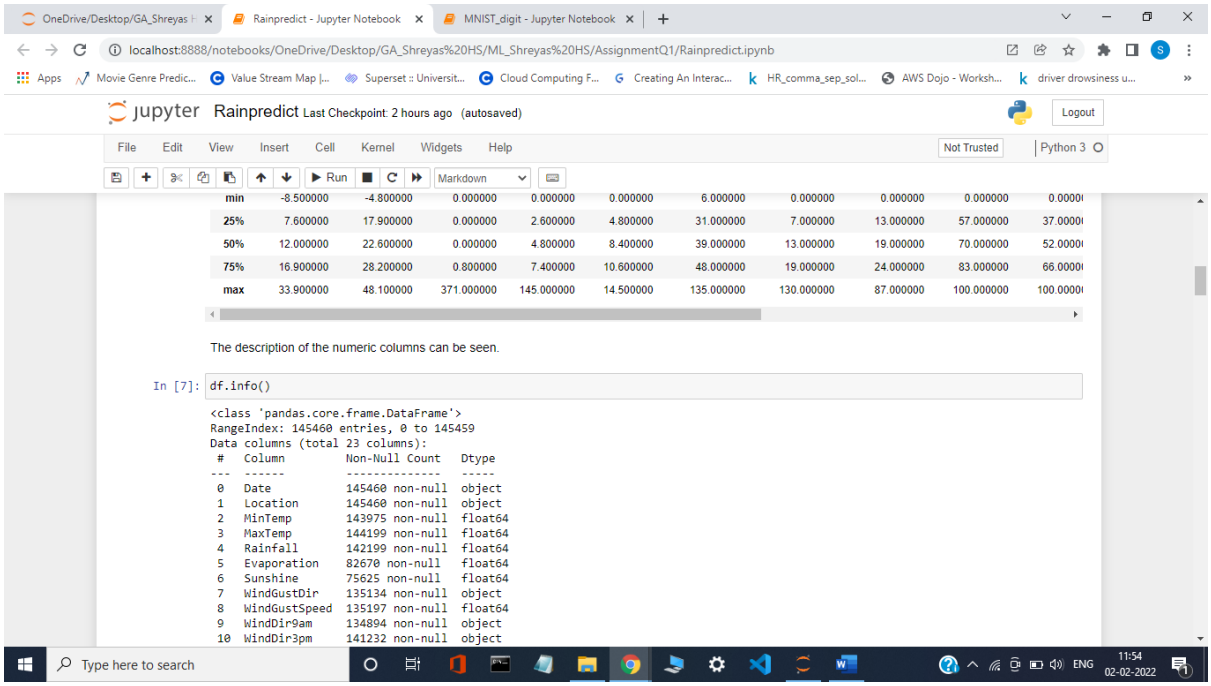
Q1. To predict if it will rain tomorrow in XYZ country using suitable **ML** approach.

Screenshots:





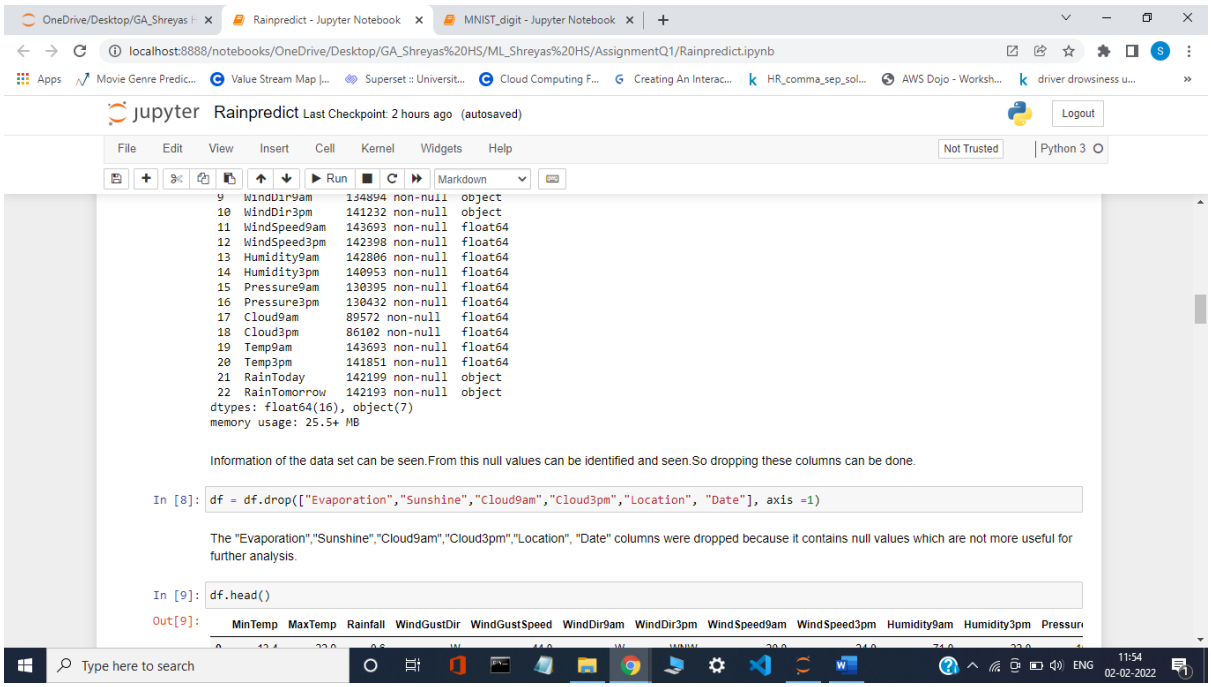


min	-8.500000	-4.800000	0.000000	0.000000	0.000000	6.000000	0.000000	0.000000	0.000000	0.0000
25%	7.600000	17.900000	0.000000	2.600000	4.800000	31.000000	7.000000	13.000000	57.000000	37.0000
50%	12.000000	22.600000	0.000000	4.800000	8.400000	39.000000	13.000000	19.000000	70.000000	52.0000
75%	16.900000	28.200000	0.800000	7.400000	10.600000	48.000000	19.000000	24.000000	83.000000	66.0000
max	33.900000	48.100000	371.000000	145.000000	14.500000	135.000000	130.000000	87.000000	100.000000	100.0000

The description of the numeric columns can be seen.

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 23 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Date           145460 non-null  object
 1   Location       145460 non-null  object
 2   MinTemp        143975 non-null  float64
 3   MaxTemp        144199 non-null  float64
 4   Rainfall       142199 non-null  float64
 5   Evaporation    82670 non-null   float64
 6   Sunshine       75625 non-null   float64
 7   WindGustDir    135134 non-null  object
 8   WindGustSpeed  135197 non-null  float64
 9   WindDir9am     134894 non-null  object
 10  WindDir3pm     141232 non-null  object
```




```
 9   WindDir9am     134894 non-null  object
 10  WindDir3pm     141232 non-null  object
 11  WindSpeed9am   143693 non-null  float64
 12  WindSpeed3pm   142398 non-null  float64
 13  Humidity9am    142806 non-null  float64
 14  Humidity3pm    140953 non-null  float64
 15  Pressure9am    130395 non-null  float64
 16  Pressure3pm    130432 non-null  float64
 17  Cloud9am       89572 non-null   float64
 18  Cloud3pm       86102 non-null   float64
 19  Temp9am        143693 non-null  float64
 20  Temp3pm        141851 non-null  float64
 21  RainToday      142199 non-null  object
 22  RainTomorrow   142193 non-null  object
dtypes: float64(16), object(7)
memory usage: 25.5+ MB
```

Information of the data set can be seen.From this null values can be identified and seen.So dropping these columns can be done.

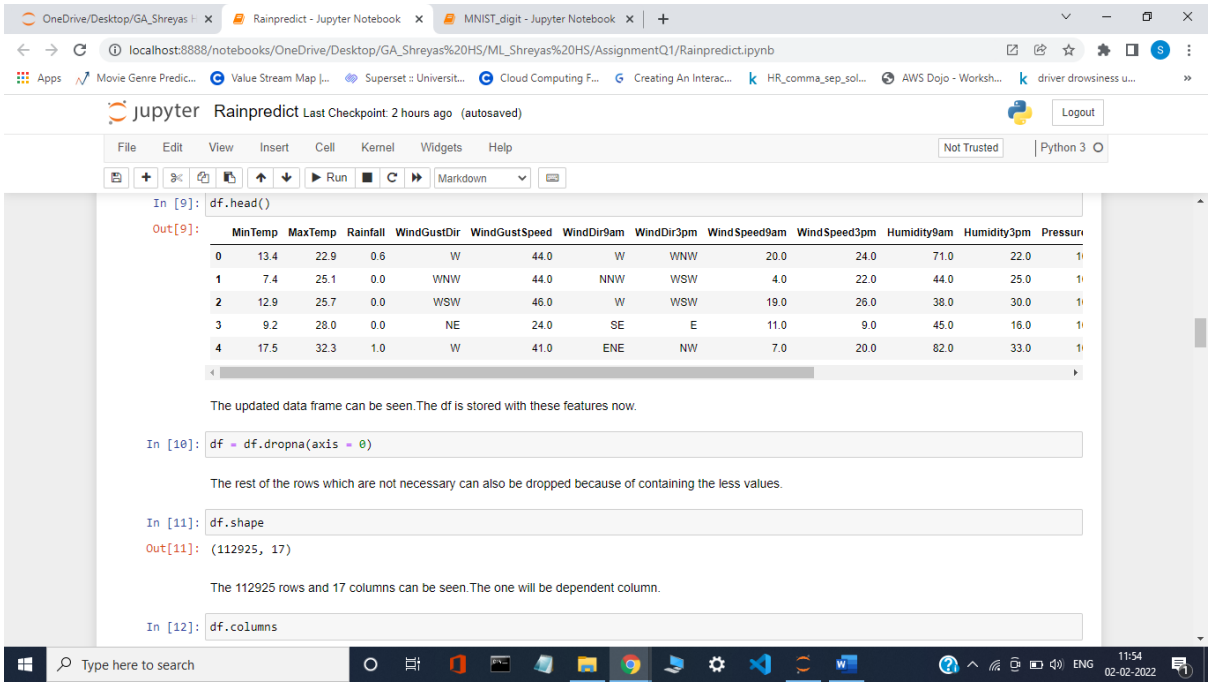
In [8]:
```
df = df.drop(["Evaporation","Sunshine","Cloud9am","Cloud3pm","Location", "Date"], axis =1)
```

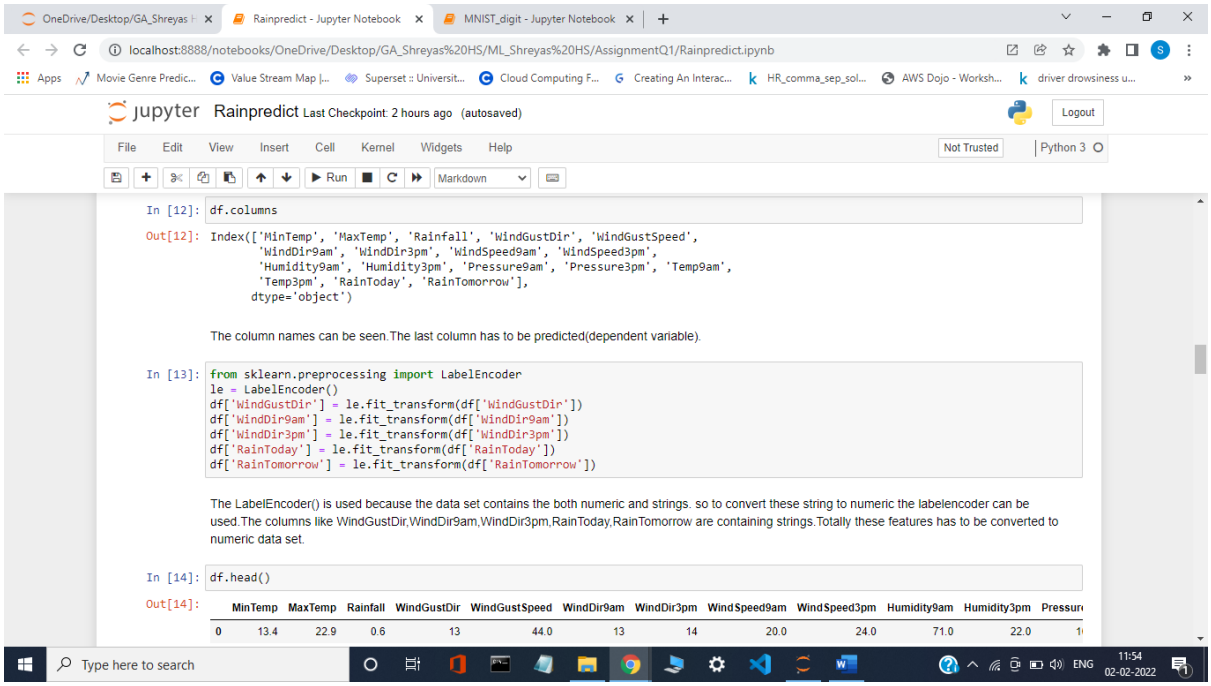
The "Evaporation","Sunshine","Cloud9am","Cloud3pm","Location", "Date" columns were dropped because it contains null values which are not more useful for further analysis.

In [9]:
```
df.head()
```

Out[9]:

	MinTemp	MaxTemp	Rainfall	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressur


OneDrive/Desktop/GA_Shreyas
Rainpredict - Jupyter Notebook
MNIST_digit - Jupyter Notebook
localhost:8888/notebooks/OneDrive/Desktop/GA_Shreyas%20HS/ML_Shreyas%20HS/AssignmentQ1/Rainpredict.ipynb
jupyter Rainpredict Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
In [9]: df.head()
Out[9]:
MinTemp MaxTemp Rainfall WindGustDir WindGustSpeed WindDir9am WindDir3pm WindSpeed9am WindSpeed3pm Humidity9am Humidity3pm Pressur
0 13.4 22.9 0.6 W 44.0 W WNW 20.0 24.0 71.0 22.0
1 7.4 25.1 0.0 WNW 44.0 NNW WSW 4.0 22.0 44.0 25.0
2 12.9 25.7 0.0 WSW 46.0 W WSW 19.0 26.0 38.0 30.0
3 9.2 28.0 0.0 NE 24.0 SE E 11.0 9.0 45.0 16.0
4 17.5 32.3 1.0 W 41.0 ENE NW 7.0 20.0 82.0 33.0
The updated data frame can be seen.The df is stored with these features now.
In [10]: df = df.dropna(axis = 0)
The rest of the rows which are not necessary can also be dropped because of containing the less values.
In [11]: df.shape
Out[11]: (112925, 17)
The 112925 rows and 17 columns can be seen.The one will be dependent column.
In [12]: df.columns
Type here to search
ENG
11:54
02-02-2022


OneDrive/Desktop/GA_Shreyas
Rainpredict - Jupyter Notebook
MNIST_digit - Jupyter Notebook
localhost:8888/notebooks/OneDrive/Desktop/GA_Shreyas%20HS/ML_Shreyas%20HS/AssignmentQ1/Rainpredict.ipynb
jupyter Rainpredict Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
In [12]: df.columns
Out[12]: Index(['MinTemp', 'MaxTemp', 'Rainfall', 'WindGustDir', 'WindGustSpeed',
'WindDir9am', 'WindDir3pm', 'WindSpeed9am', 'WindSpeed3pm',
'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Temp9am',
'Temp3pm', 'RainToday', 'RainTomorrow'],
dtype='object')
The column names can be seen.The last column has to be predicted(dependent variable).
In [13]: from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['WindGustDir'] = le.fit_transform(df['WindGustDir'])
df['WindDir9am'] = le.fit_transform(df['WindDir9am'])
df['WindDir3pm'] = le.fit_transform(df['WindDir3pm'])
df['RainToday'] = le.fit_transform(df['RainToday'])
df['RainTomorrow'] = le.fit_transform(df['RainTomorrow'])
The LabelEncoder() is used because the data set contains the both numeric and strings. so to convert these string to numeric the labelencoder can be used.The columns like WindGustDir,WindDir9am,WindDir3pm,RainToday,RainTomorrow are containing strings.Totally these features has to be converted to numeric data set.
In [14]: df.head()
Out[14]:
MinTemp MaxTemp Rainfall WindGustDir WindGustSpeed WindDir9am WindDir3pm WindSpeed9am WindSpeed3pm Humidity9am Humidity3pm Pressur
0 13.4 22.9 0.6 13 44.0 13 14 20.0 24.0 71.0 22.0
Type here to search
ENG
11:54
02-02-2022

OneDrive/Desktop/GA_Shreyas | Rainpredict - Jupyter Notebook | MNIST_digit - Jupyter Notebook

localhost:8888/notebooks/OneDrive/Desktop/GA_Shreyas%20HS/ML_Shreyas%20HS/AssignmentQ1/Rainpredict.ipynb

Apps | Movie Genre Predic... | Value Stream Map |... | Superset :: Universit... | Cloud Computing F... | Creating An Interac... | HR_comma_sep_sol... | AWS Dojo - Worksh... | driver drowsiness u...

jupyter **Rainpredict** Last Checkpoint: 2 hours ago (autosaved) — Logout

File Edit View Insert Cell Kernel Widgets Help — Not Trusted — Python 3

Run | Markdown

In [14]:
```
df.head()
```

Out[14]:

	MinTemp	MaxTemp	Rainfall	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressur
0	13.4	22.9	0.6	13	44.0	13	14	20.0	24.0	71.0	22.0	1
1	7.4	25.1	0.0	14	44.0	6	15	4.0	22.0	44.0	25.0	1
2	12.9	25.7	0.0	15	46.0	13	15	19.0	26.0	38.0	30.0	1
3	9.2	28.0	0.0	4	24.0	9	0	11.0	9.0	45.0	16.0	1
4	17.5	32.3	1.0	13	41.0	1	7	7.0	20.0	82.0	33.0	1

So all string data set were transformed to numeric data.

In [15]:
```
x = df.drop(['RainTomorrow'], axis = 1)
y = df['RainTomorrow']
```

Dividing the dataset into dependent and independent columns."x" contains independent columns.The dropping of RainTomorrow becasue it has to be predicted and assigning that after dropping to "y" that is updating the "x".

In [16]:
```
x.head()
```

Out[16]:

	MinTemp	MaxTemp	Rainfall	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressur
0	13.4	22.9	0.6	13	44.0	13	14	20.0	24.0	71.0	22.0	1
1	7.4	25.1	0.0	14	44.0	6	15	4.0	22.0	44.0	25.0	1

Type here to search — ENG 11:54 02-02-2022

OneDrive/Desktop/GA_Shreyas | Rainpredict - Jupyter Notebook | MNIST_digit - Jupyter Notebook

localhost:8888/notebooks/OneDrive/Desktop/GA_Shreyas%20HS/ML_Shreyas%20HS/AssignmentQ1/Rainpredict.ipynb

Apps | Movie Genre Predic... | Value Stream Map |... | Superset :: Universit... | Cloud Computing F... | Creating An Interac... | HR_comma_sep_sol... | AWS Dojo - Worksh... | driver drowsiness u...

jupyter **Rainpredict** Last Checkpoint: 2 hours ago (autosaved) — Logout

File Edit View Insert Cell Kernel Widgets Help — Not Trusted — Python 3

Run | Markdown

In [16]:
```
x.head()
```

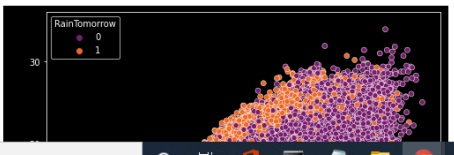
Out[16]:

	MinTemp	MaxTemp	Rainfall	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressur
0	13.4	22.9	0.6	13	44.0	13	14	20.0	24.0	71.0	22.0	1
1	7.4	25.1	0.0	14	44.0	6	15	4.0	22.0	44.0	25.0	1
2	12.9	25.7	0.0	15	46.0	13	15	19.0	26.0	38.0	30.0	1
3	9.2	28.0	0.0	4	24.0	9	0	11.0	9.0	45.0	16.0	1
4	17.5	32.3	1.0	13	41.0	1	7	7.0	20.0	82.0	33.0	1

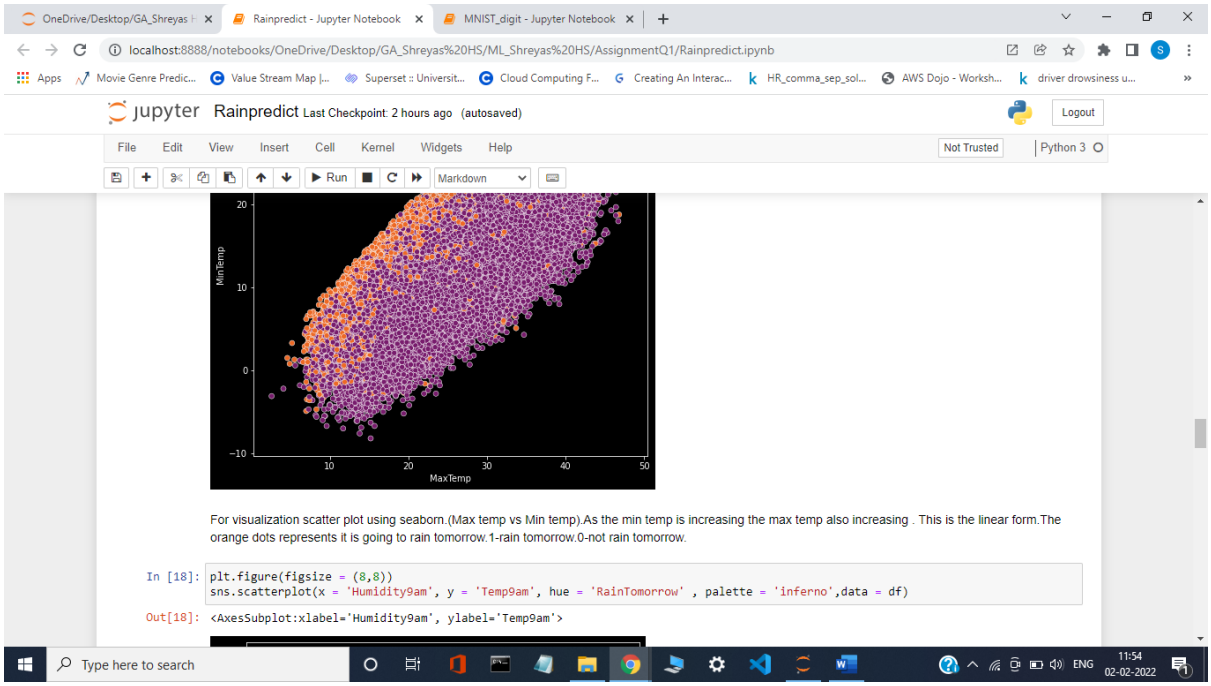
Checking the head part shows the last column is dropped.

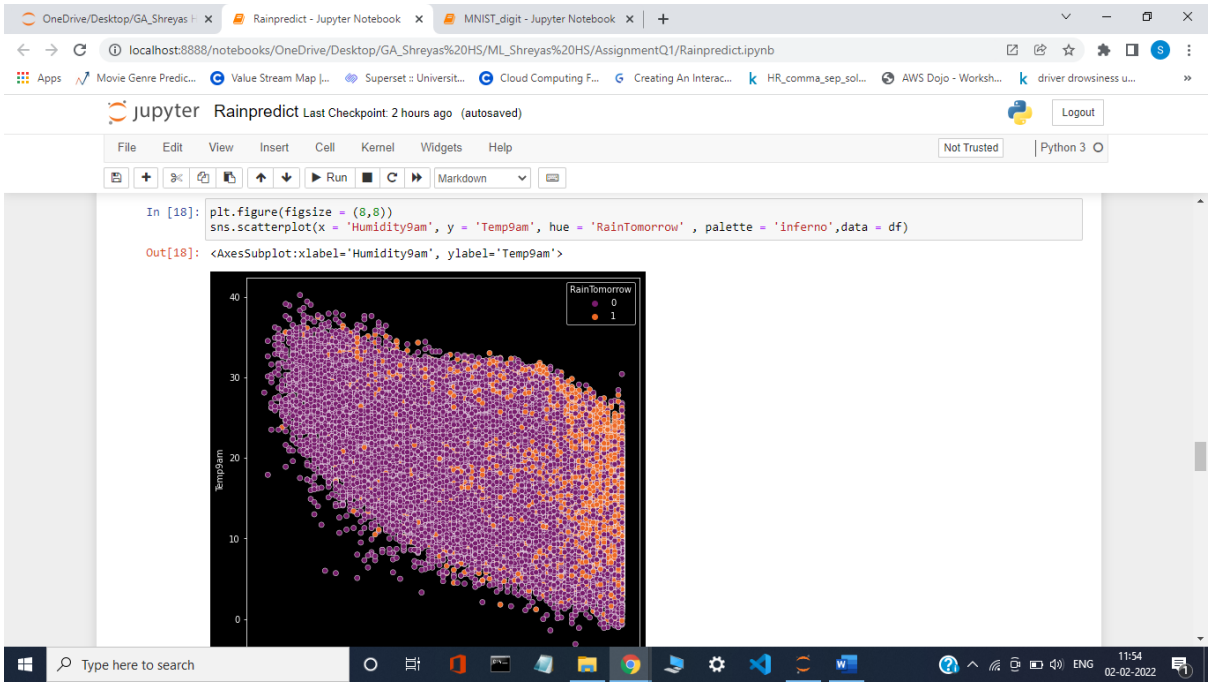
In [17]:
```
plt.figure(figsize = (8,8))
sns.scatterplot(x = 'MaxTemp', y = 'MinTemp', hue = 'RainTomorrow' , palette = 'inferno',data = df)
```

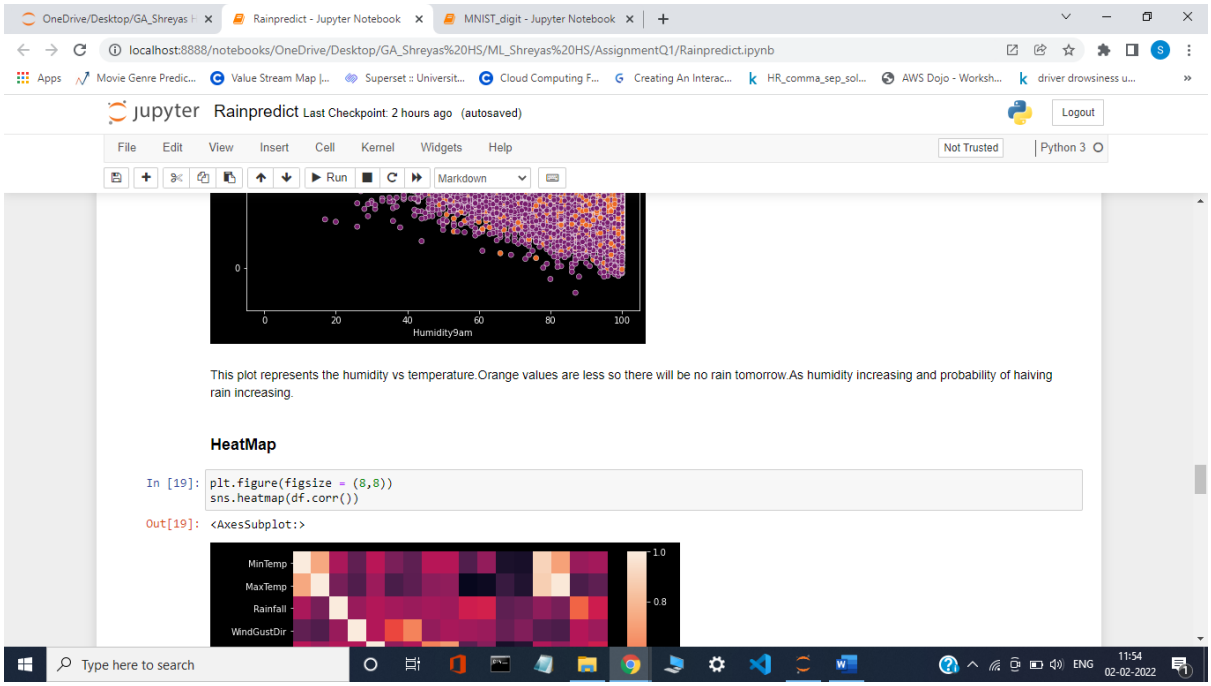
Out[17]:
```
<AxesSubplot:xlabel='MaxTemp', ylabel='MinTemp'>
```



Type here to search — ENG 11:54 02-02-2022


OneDrive/Desktop/GA_Shreyas
Rainpredict - Jupyter Notebook
MNIST_digit - Jupyter Notebook
localhost:8888/notebooks/OneDrive/Desktop/GA_Shreyas%20HS/ML_Shreyas%20HS/AssignmentQ1/Rainpredict.ipynb
jupyter Rainpredict Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
MinTemp
MaxTemp
20
10
0
−10
10
20
30
40
50
For visualization scatter plot using seaborn.(Max temp vs Min temp).As the min temp is increasing the max temp also increasing . This is the linear form.The orange dots represents it is going to rain tomorrow.1-rain tomorrow.0-not rain tomorrow.
In [18]:
plt.figure(figsize = (8,8))
sns.scatterplot(x = 'Humidity9am', y = 'Temp9am', hue = 'RainTomorrow' , palette = 'inferno',data = df)
Out[18]:
<AxesSubplot:xlabel='Humidity9am', ylabel='Temp9am'>
Type here to search
ENG
11:54
02-02-2022


OneDrive/Desktop/GA_Shreyas
Rainpredict - Jupyter Notebook
MNIST_digit - Jupyter Notebook
localhost:8888/notebooks/OneDrive/Desktop/GA_Shreyas%20HS/ML_Shreyas%20HS/AssignmentQ1/Rainpredict.ipynb
jupyter Rainpredict Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
In [18]:
plt.figure(figsize = (8,8))
sns.scatterplot(x = 'Humidity9am', y = 'Temp9am', hue = 'RainTomorrow' , palette = 'inferno',data = df)
Out[18]:
<AxesSubplot:xlabel='Humidity9am', ylabel='Temp9am'>
RainTomorrow
0
1
Temp9am
40
30
20
10
0
Type here to search
ENG
11:54
02-02-2022

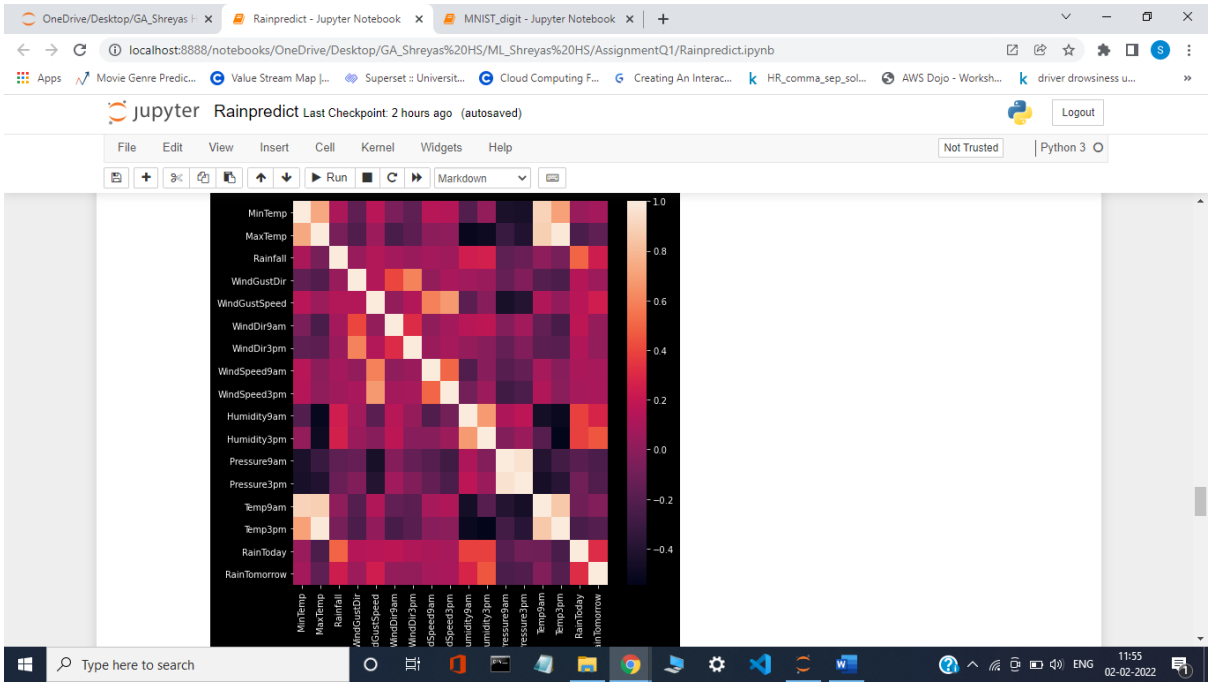


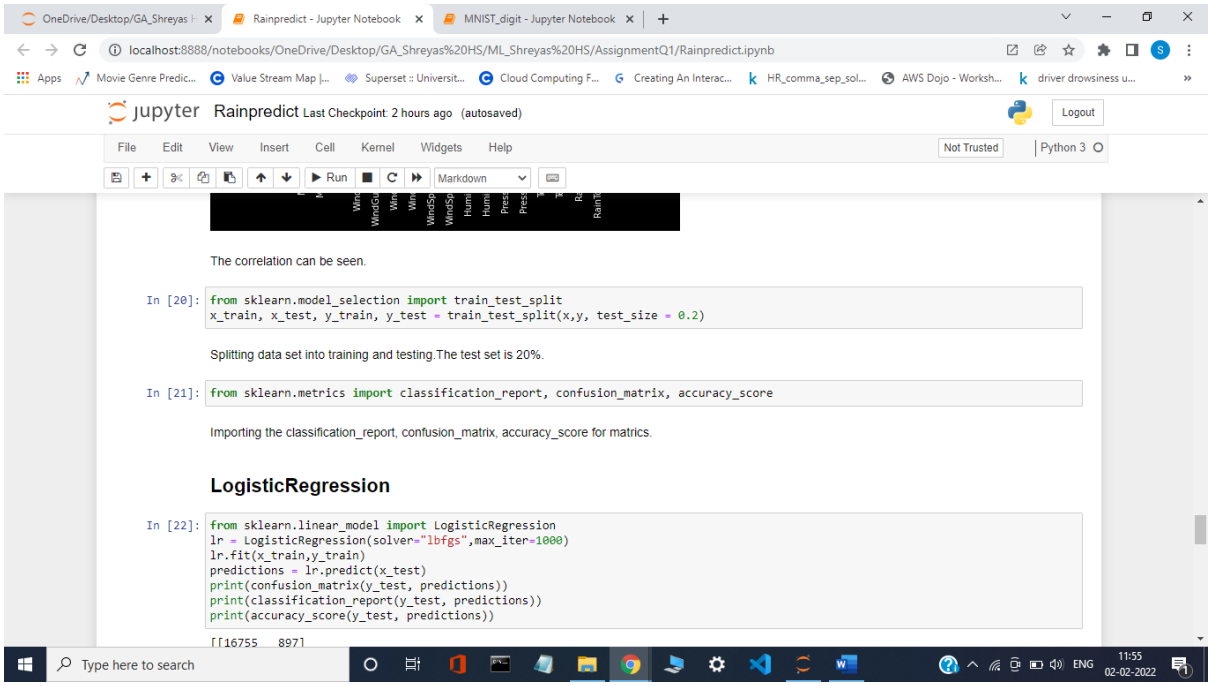
This plot represents the humidity vs temperature.Orange values are less so there will be no rain tomorrow.As humidity increasing and probability of haiving rain increasing.

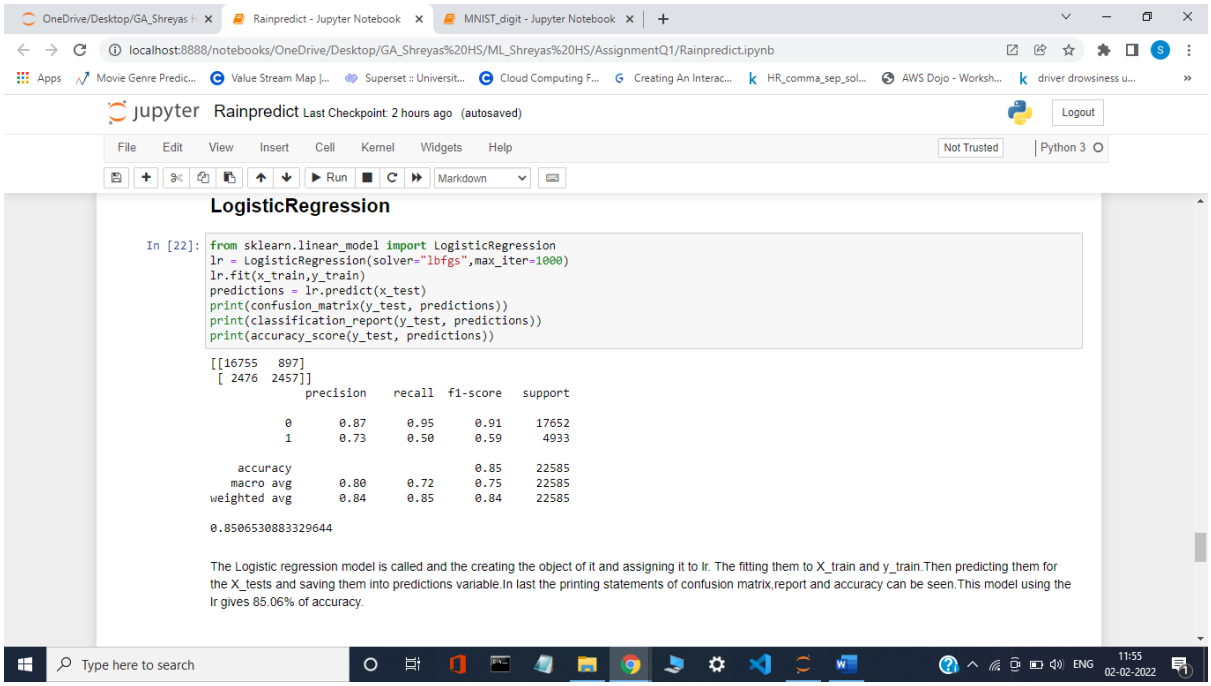
HeatMap

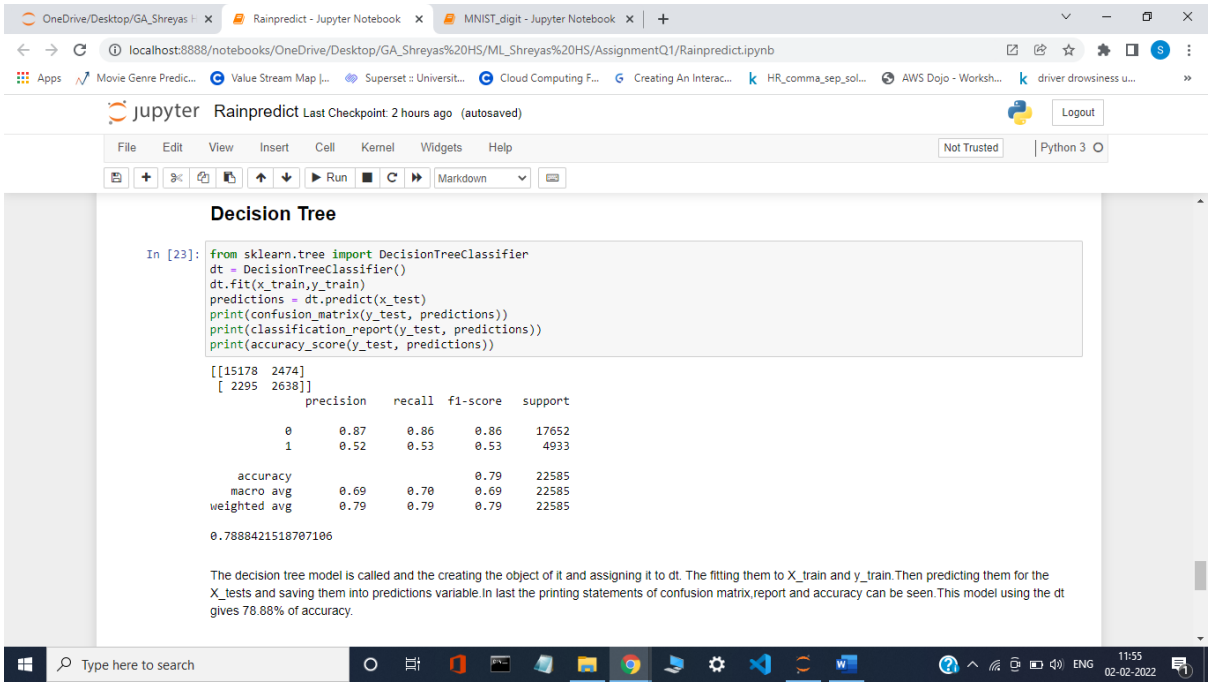
In [19]:
```
plt.figure(figsize = (8,8))
sns.heatmap(df.corr())
```

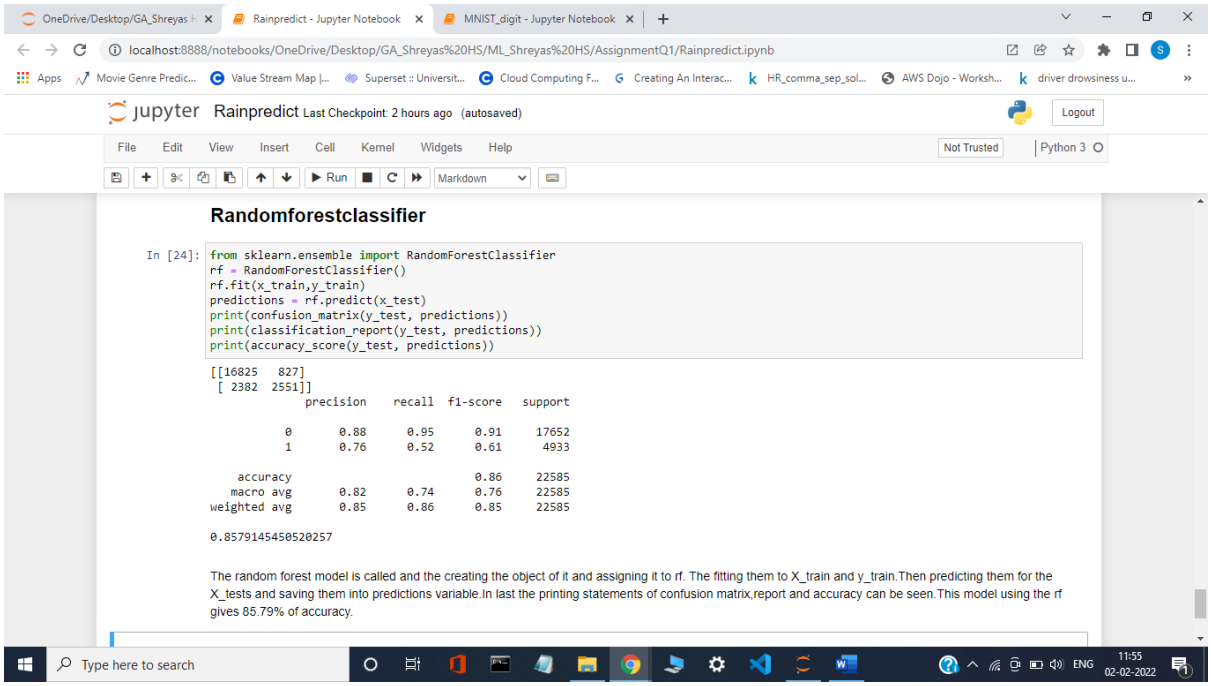
Out[19]: <AxesSubplot:>




OneDrive/Desktop/GA_Shreyas
Rainpredict - Jupyter Notebook
MNIST_digit - Jupyter Notebook
localhost:8888/notebooks/OneDrive/Desktop/GA_Shreyas%20HS/ML_Shreyas%20HS/AssignmentQ1/Rainpredict.ipynb
jupyter Rainpredict Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
The correlation can be seen.
In [20]:
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2)
Splitting data set into training and testing.The test set is 20%.
In [21]:
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
Importing the classification_report, confusion_matrix, accuracy_score for matrics.
LogisticRegression
In [22]:
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(solver="lbfgs",max_iter=1000)
lr.fit(x_train,y_train)
predictions = lr.predict(x_test)
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
[[16755 897]
Type here to search
ENG
11:55
02-02-2022


OneDrive/Desktop/GA_Shreyas
Rainpredict - Jupyter Notebook
MNIST_digit - Jupyter Notebook
localhost:8888/notebooks/OneDrive/Desktop/GA_Shreyas%20HS/ML_Shreyas%20HS/AssignmentQ1/Rainpredict.ipynb
jupyter Rainpredict Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
LogisticRegression
In [22]:
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(solver="lbfgs",max_iter=1000)
lr.fit(x_train,y_train)
predictions = lr.predict(x_test)
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
[[16755 897]
[2476 2457]]
precision recall f1-score support
0 0.87 0.95 0.91 17652
1 0.73 0.50 0.59 4933
accuracy 0.85 22585
macro avg 0.80 0.72 0.75 22585
weighted avg 0.84 0.85 0.84 22585
0.8506530883329644
The Logistic regression model is called and the creating the object of it and assigning it to lr. The fitting them to X_train and y_train.Then predicting them for the X_tests and saving them into predictions variable.In last the printing statements of confusion matrix,report and accuracy can be seen.This model using the lr gives 85.06% of accuracy.
Type here to search
ENG
11:55
02-02-2022


Rainpredict - Jupyter Notebook
MNIST_digit - Jupyter Notebook
localhost:8888/notebooks/OneDrive/Desktop/GA_Shreyas%20HS/ML_Shreyas%20HS/AssignmentQ1/Rainpredict.ipynb
jupyter Rainpredict Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
Decision Tree
In [23]:
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
predictions = dt.predict(x_test)
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
[[15178 2474]
[2295 2638]]
precision recall f1-score support
0 0.87 0.86 0.86 17652
1 0.52 0.53 0.53 4933
accuracy 0.79 22585
macro avg 0.69 0.70 0.69 22585
weighted avg 0.79 0.79 0.79 22585
0.7888421518707106
The decision tree model is called and the creating the object of it and assigning it to dt. The fitting them to X_train and y_train.Then predicting them for the X_tests and saving them into predictions variable.In last the printing statements of confusion matrix,report and accuracy can be seen.This model using the dt gives 78.88% of accuracy.
Type here to search
ENG
11:55
02-02-2022


Rainpredict - Jupyter Notebook
MNIST_digit - Jupyter Notebook
localhost:8888/notebooks/OneDrive/Desktop/GA_Shreyas%20HS/ML_Shreyas%20HS/AssignmentQ1/Rainpredict.ipynb
jupyter Rainpredict Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown
Randomforestclassifier
In [24]:
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(x_train,y_train)
predictions = rf.predict(x_test)
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
[[16825 827]
[2382 2551]]
precision recall f1-score support
0 0.88 0.95 0.91 17652
1 0.76 0.52 0.61 4933
accuracy 0.86 22585
macro avg 0.82 0.74 0.76 22585
weighted avg 0.85 0.86 0.85 22585
0.8579145450520257
The random forest model is called and the creating the object of it and assigning it to rf. The fitting them to X_train and y_train.Then predicting them for the X_tests and saving them into predictions variable.In last the printing statements of confusion matrix,report and accuracy can be seen.This model using the rf gives 85.79% of accuracy.
Type here to search
ENG
11:55
02-02-2022

Code: and Description:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

These are the import statements. The pandas is python library for data analysis, numpy for mathematical operations, matplotlib for data visualization, seaborn is a library for making statistical graphics in Python.

```
plt.style.use('dark_background')
```

For background colour in the plots.

```
df = pd.read_csv('weatherAUS.csv')
```

Reading csv data from the excel sheet(.xsl) file of "weatherAus.xsl" file.And storing them in the variable called df.

```
df.head()
```

This shows the 5 rows of the data set.The features like date,location,mintemp,maxtemp,rainfall upto raintomorrow can be seen.There are 23 columns.The rain tomorrow column has to be predict so this will become "y" i.e dependent variable.Rest all were independent variables except last column.

```
df.shape
```

That is the data set contains the 145460 rows and 23 columns.

```
df.describe()
```

The description of the numeric columns can be seen.

```
df.info()
```

Information of the data set can be seen.From this null values can be identified and seen.So dropping these columns can be done.

```
df = df.drop(["Evaporation","Sunshine","Cloud9am","Cloud3pm","Location", "Date"], axis =1)
```

The "Evaporation","Sunshine","Cloud9am","Cloud3pm","Location", "Date" columns were dropped because it contains null values which are not more useful for further analysis.

```
df.head()
```

The updated data frame can be seen.The df is stored with these features now.

```
df = df.dropna(axis = 0)
```

The rest of the rows which are not necessary can also be dropped because of containing the less values.

```
df.shape
```

The 112925 rows and 17 columns can be seen.The one will be dependent column.

```
df.columns
```

The column names can be seen.The last column has to be predicted(dependent variable).

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['WindGustDir'] = le.fit_transform(df['WindGustDir'])
df['WindDir9am'] = le.fit_transform(df['WindDir9am'])
df['WindDir3pm'] = le.fit_transform(df['WindDir3pm'])
df['RainToday'] = le.fit_transform(df['RainToday'])
df['RainTomorrow'] = le.fit_transform(df['RainTomorrow'])
```

The LabelEncoder() is used because the data set contains the both numeric and strings. so to convert these string to numeric the labelencoder can be used.The columns like WindGustDir,WindDir9am,WindDir3pm,RainToday,RainTomorrow are containing strings.Totally these features has to be converted to numeric data set.

```
df.head()
```

So all string data set were transformed to numeric data.

```
x = df.drop(['RainTomorrow'], axis = 1)

y = df['RainTomorrow']
```

Dividing the dataset into dependent and independent columns."x" contains independent columns.The dropping of RainTomorrow becasue it has to be predicted and assigning that after dropping to "y" that is updating the "x".

```
x.head()
```

Checking the head part shows the last column is dropped.

```
plt.figure(figsize = (8,8))

sns.scatterplot(x = 'MaxTemp', y = 'MinTemp', hue = 'RainTomorrow' , palette = 'inferno',data = df)
```

For visualization scatter plot using seaborn.(Max temp vs Min temp).As the min temp is increasing the max temp also increasing . This is the linear form.The orange dots represents it is going to rain tomorrow.1-rain tomorrow.0-not rain tomorrow.

```
plt.figure(figsize = (8,8))

sns.scatterplot(x = 'Humidity9am', y = 'Temp9am', hue = 'RainTomorrow' , palette = 'inferno',data = df)
```

This plot represents the humidity vs temperature.Orange values are less so there will be no rain tomorrow.As humidity increasing and probability of haiving rain increasing.

HeatMap

```
plt.figure(figsize = (8,8))

sns.heatmap(df.corr())
```

The correlation can be seen.

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2)
```

Splitting data set into training and testing.The test set is 20%.

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

Importing the classification_report, confusion_matrix, accuracy_score for matrics.

LogisticRegression

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(solver="lbfgs",max_iter=1000)
lr.fit(x_train,y_train)
predictions = lr.predict(x_test)
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

The Logistic regression model is called and the creating the object of it and assigning it to lr. The fitting them to X_train and y_train.Then predicting them for the X_tests and saving them into predictions variable.In last the printing statements of confusion matrix,report and accuracy can be seen.This model using the lr gives 85.06% of accuracy.

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(x_train,y_train)
predictions = dt.predict(x_test)
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print(accuracy_score(y_test, predictions))
```

The decision tree model is called and the creating the object of it and assigning it to dt. The fitting them to X_train and y_train.Then predicting them for the X_tests and saving them into predictions variable.In last the printing statements of confusion matrix,report and accuracy can be seen.This model using the dt gives 78.88% of accuracy.

Randomforestclassifier

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()

rf.fit(x_train,y_train)

predictions = rf.predict(x_test)

print(confusion_matrix(y_test, predictions))

print(classification_report(y_test, predictions))

print(accuracy_score(y_test, predictions))
```

The random forest model is called and the creating the object of it and assigning it to rf. The fitting them to X_train and y_train.Then predicting them for the X_tests and saving them into predictions variable.In last the printing statements of confusion matrix,report and accuracy can be seen.This model using the rf gives 85.79% of accuracy.

Inference:

So the data analysis says that for this data set the **random forest classifer** with 85.79% gives the better accuracy compare to the logistic regression and decision tree.