

### 

### 

### GOUP MEMBERS: 1-Syeda Rameen Ezam(P19101073) 2-Alvina binte waqar(P19101005) COURSE INCHARGE:

### Dr Tehseen Ahmed Jilani

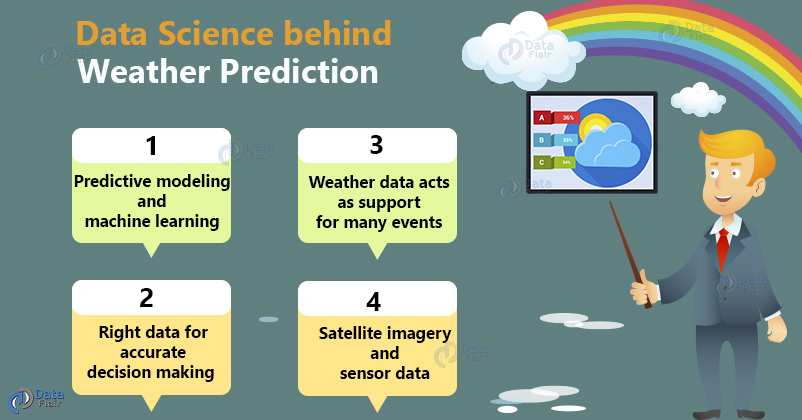
### COURSE:

### Data Warehousing and Data Mining CLASS MCS(Final)

REPORT:  
**INTRODUCTION**:  
It cannot be denied that weather forecasting, i.e. predicting weather behavior, is a very challenging task, even with the rapid growth in science. Weather is known to be in the area of meteorology. The process is carried out by collecting data related to the current state of weather like rain, heat, wind, and fog. Data mining techniques in this field has increasingly developed over the last ten years.

Weather forecast is the application of science to predict the condition of atmosphere for a given city or area. It is made by collecting a maximum amount of data possible about the current condition of the atmosphere and using the understanding of atmospheric processes to determine how the atmosphere involve in the future.

Weather forecast involves three steps:   
1-Observation  
2-Analysis

3-Extrapolation   
  
  
  
Many businesses are depend on weather conditions for example agriculture activities, Construction work, Airport control authority etc.

### ADVANTAGES OF WEATHER FORECASTING:

Advantages of weather forecasting are:

* People are warned prior to what the weather will be like on a particular day.
* To help people take proper precautions to secure themselves and their families in case of unwanted occurrences.
* Organizations can work better with the help of accurate weather predictions.
* It helps to deliver visual forecasts by various methods that most companies prefer.
* Weather forecasting highly benefits the agriculture sector for buying/selling livestock.
* It also assists the farmers to decide when to plant crops, pastures, and when to irrigate.
* It is the best method for management of inventory, selling strategies and crop forecasts.
* It provides the business with valuable information that the business can use to make  
  decisions about future business strategies.

**LITERATURE REWIEW:**

**ABOUT DATASET:**

This dataset contains about 10 years of daily weather observations from numerous Australian weather stations. RainTomorrow is the target variable to predict. It means -- did it rain the next day, Yes or No?

There are 23 attributes in this dataset.

**1-IMPORTING GENERAL PROPERTIES AND OUR DATASET:**

import seaborn as sns

import warnings

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

%matplotlib inline

# ignore all warnings

warnings. filterwarnings("ignore")

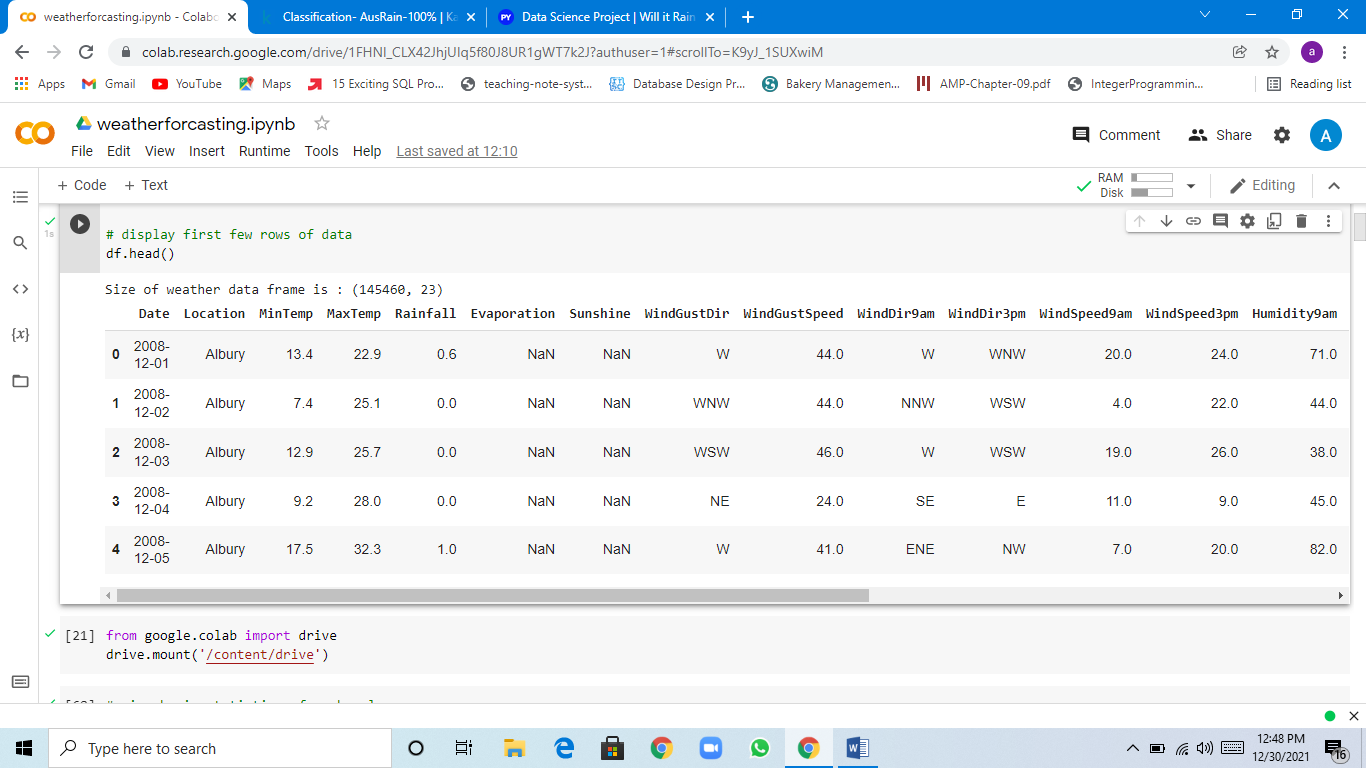
# read data into dataframe variable df

df = pd.read\_csv("/content/weatherAUS.csv")

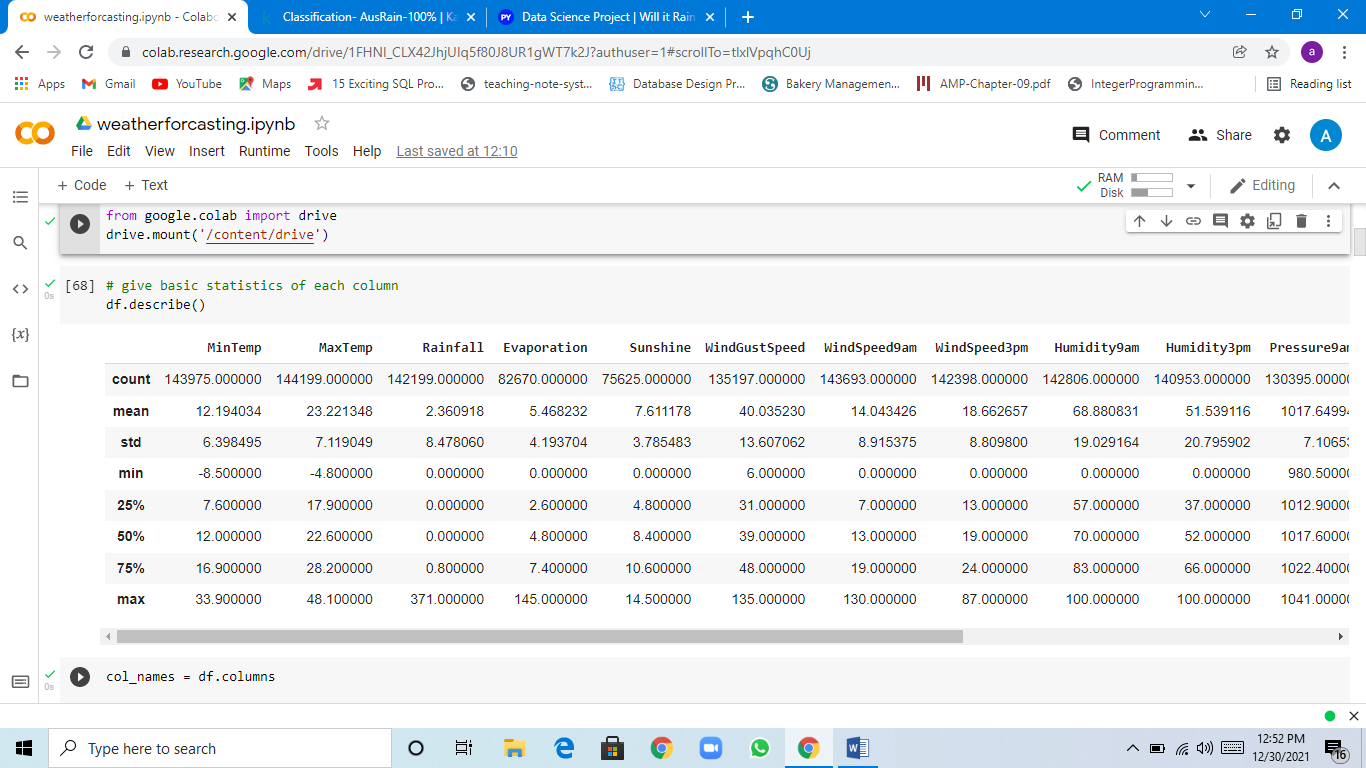
print('Size of weather data frame is :', df.shape)

# display first few rows of data

df.head()



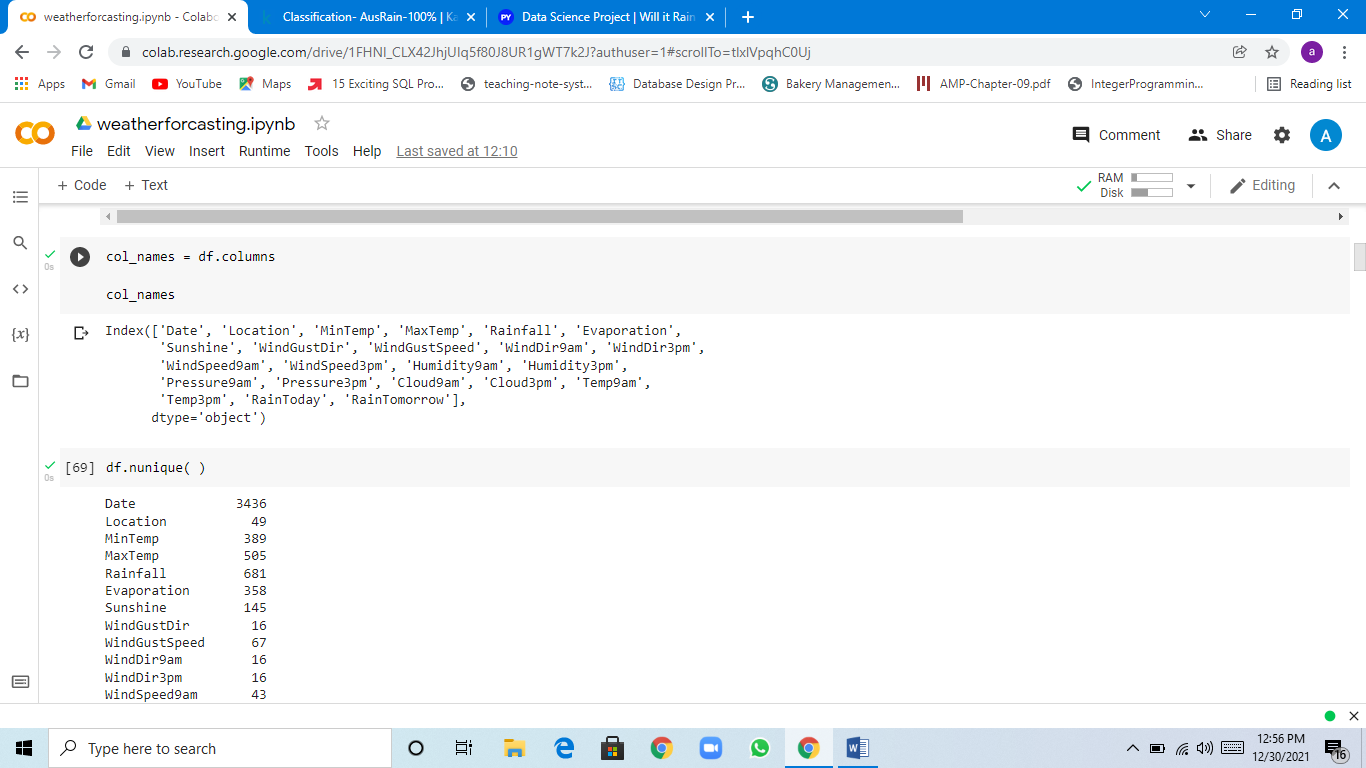
**2-STATISTICS FOR ALL THE COLUMNS OF THE DATASET:**

df.describe()

**COLUMNS NAME IN DATASET:**

col\_names = df.columns

col\_names

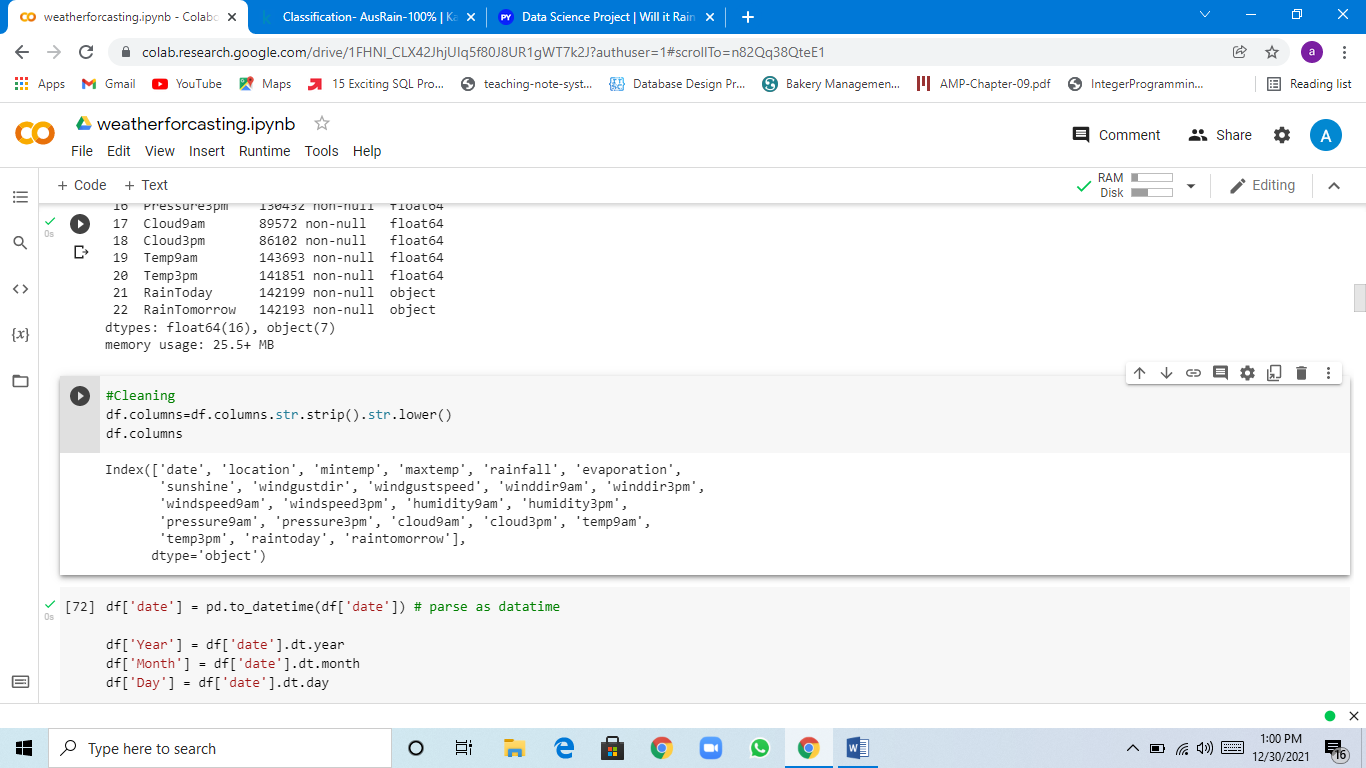


**3-CLEANING THE DATA:**

CONVERTING IN LOWERCASE:

df.columns=df.columns.str.strip().str.lower()

df.columns



CHANGE DATE FORMAT:

df['date'] = pd.to\_datetime(df['date']) # parse as datatime

df['Year'] = df['date'].dt.year

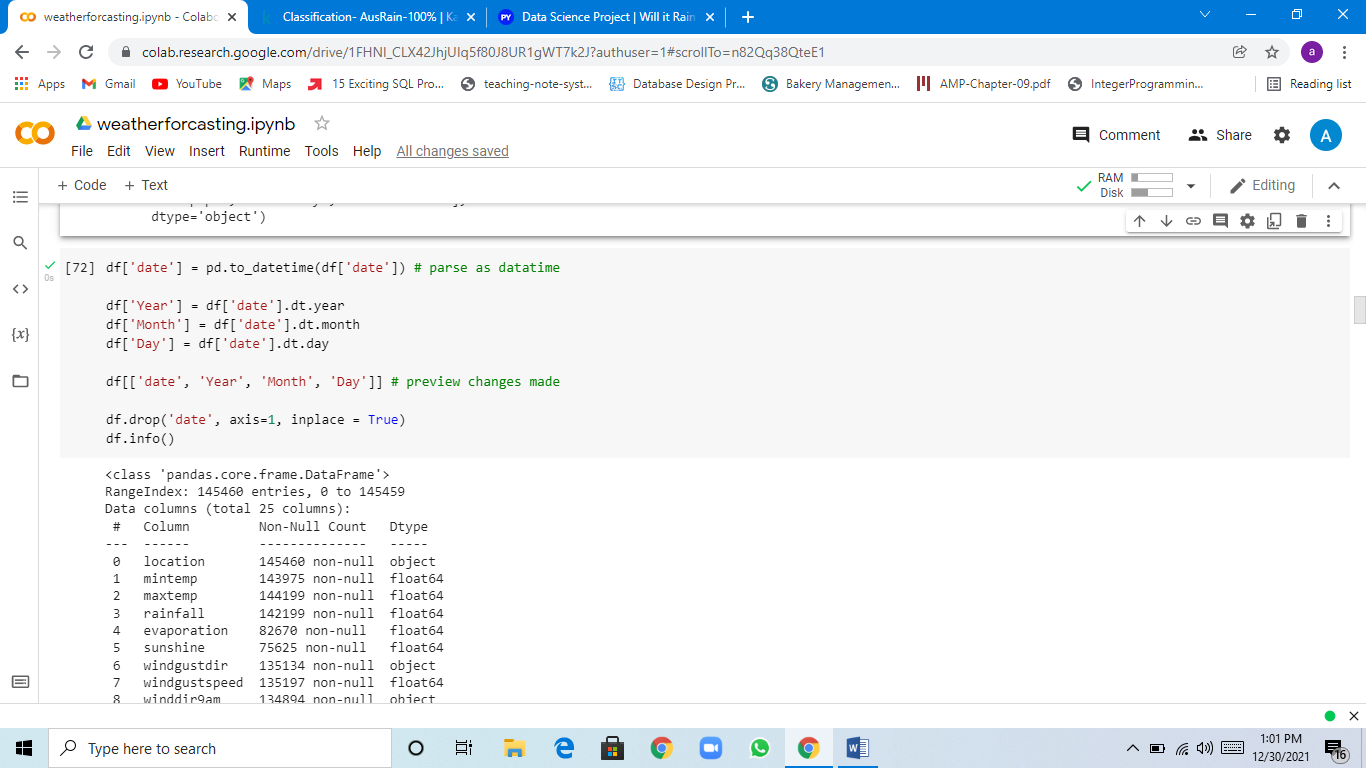
df['Month'] = df['date'].dt.month

df['Day'] = df['date'].dt.day

df[['date', 'Year', 'Month', 'Day']] # preview changes made

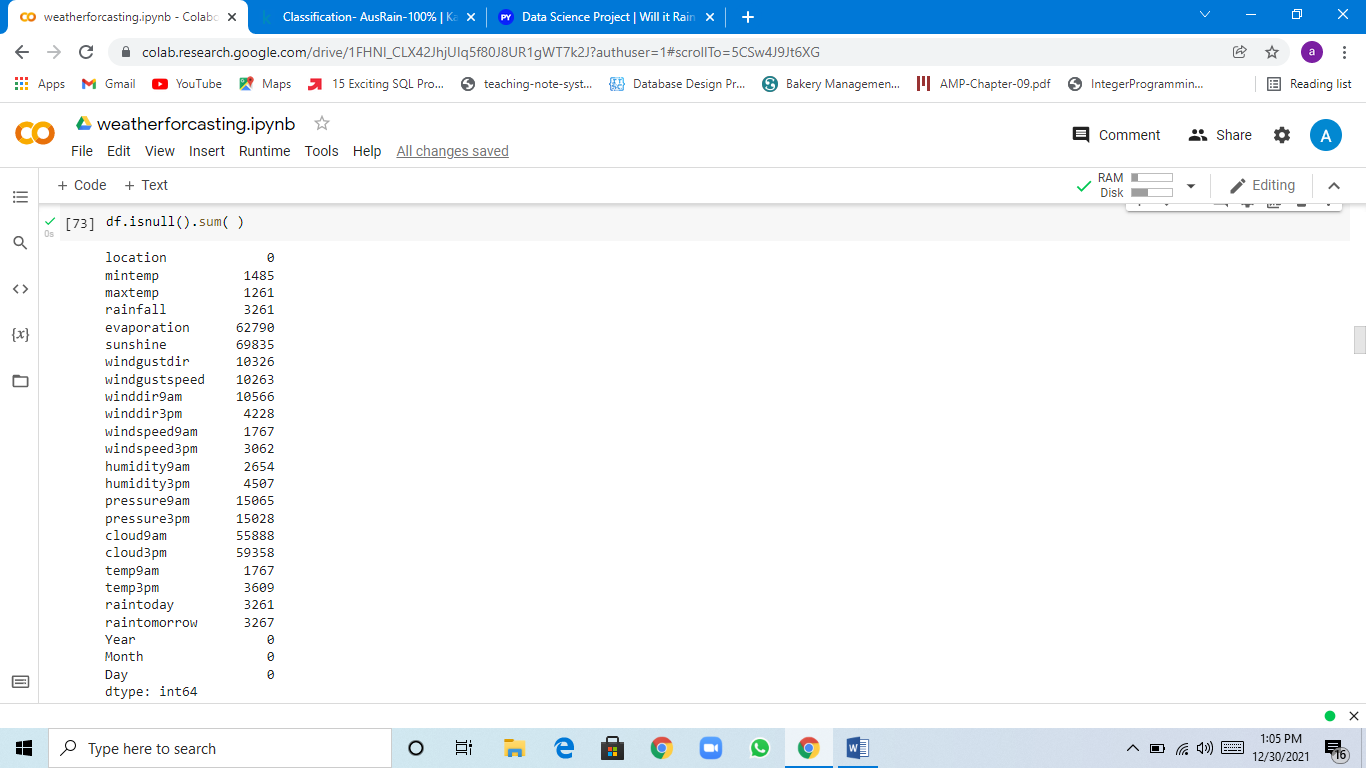
df.drop('date', axis=1, inplace = True)

df.info()



DEAL WITH NULL OR MISSING VALUES:

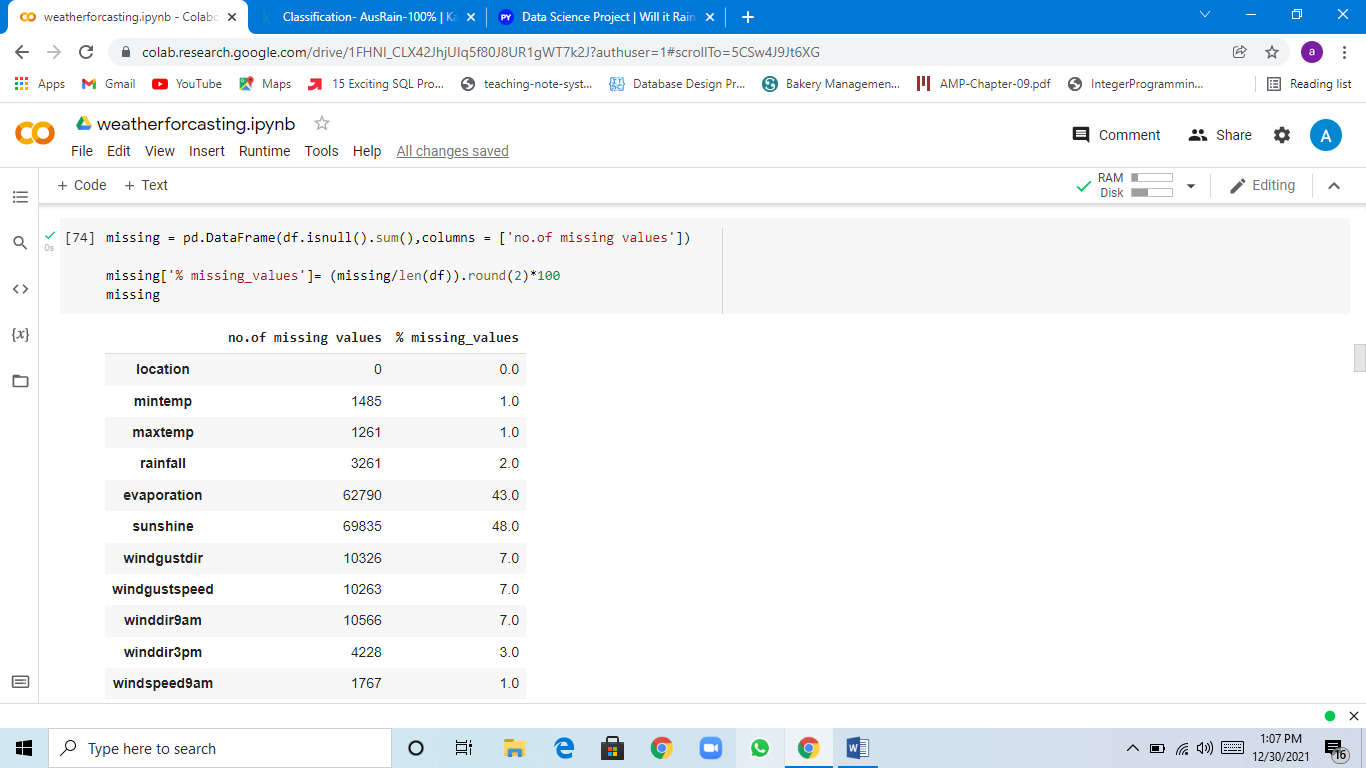
df.isnull().sum( )



missing = pd.DataFrame(df.isnull().sum(),columns = ['no.of missing values'])

missing['% missing\_values']= (missing/len(df)).round(2)\*100

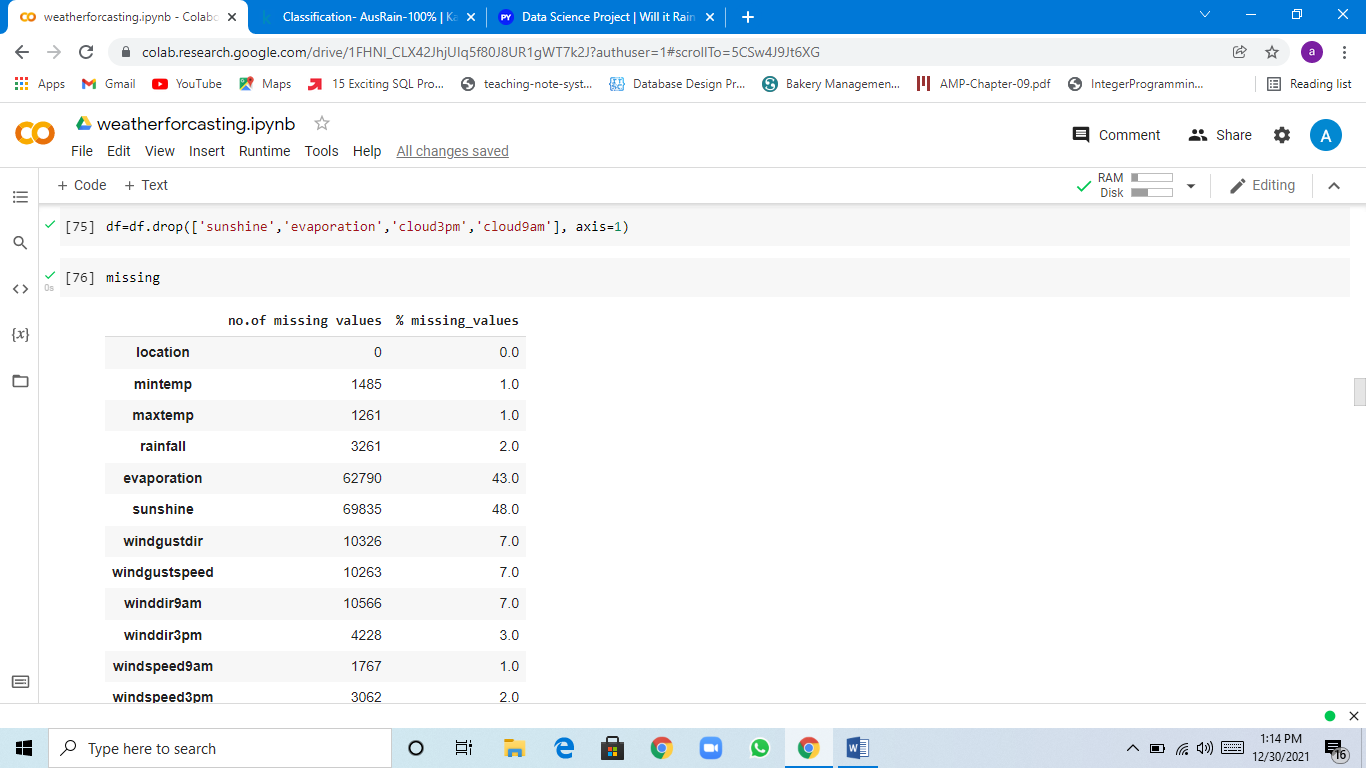
missing



DELETE COLUMNS WITH MORE MISSING VALUES:

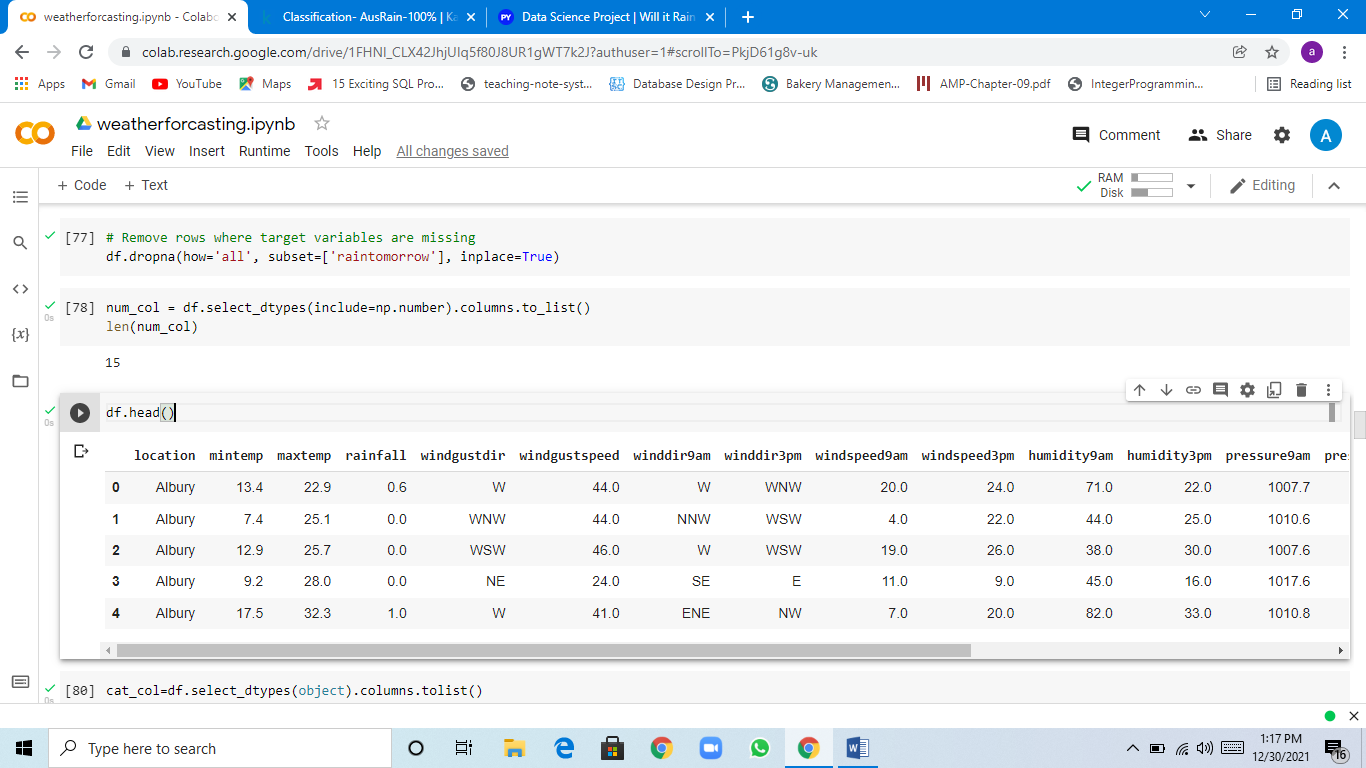
df=df.drop(['sunshine','evaporation','cloud3pm','cloud9am'], axis=1)

missing



# Remove rows where target variables are missing

df.dropna(how='all', subset=['raintomorrow'], inplace=True)



SORTING DATA INTO NUMERICAL AND CATEGORICAL FEATURES:

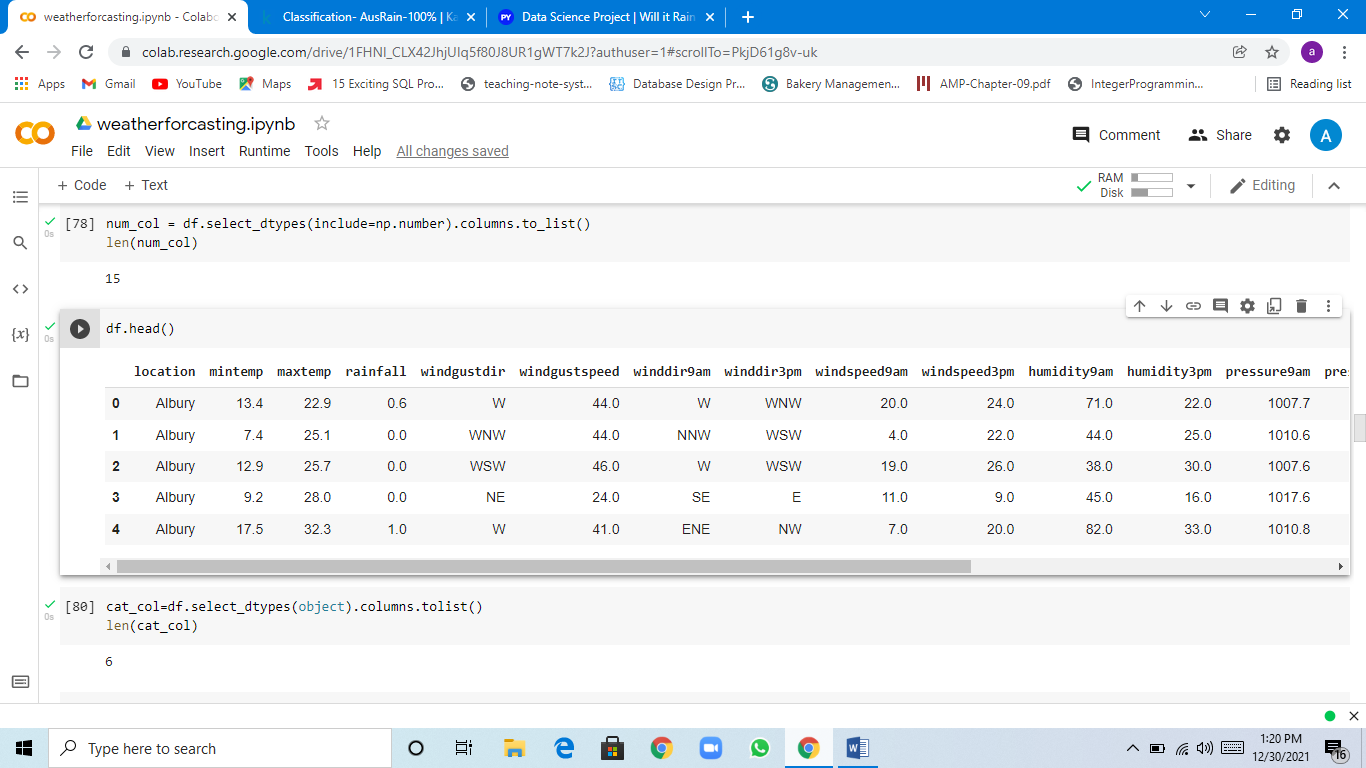
num\_col = df.select\_dtypes(include=np.number).columns.to\_list()

len(num\_col)

df.head()

cat\_col=df.select\_dtypes(object).columns.tolist()

len(cat\_col)



NUMERICAL FEATURES:

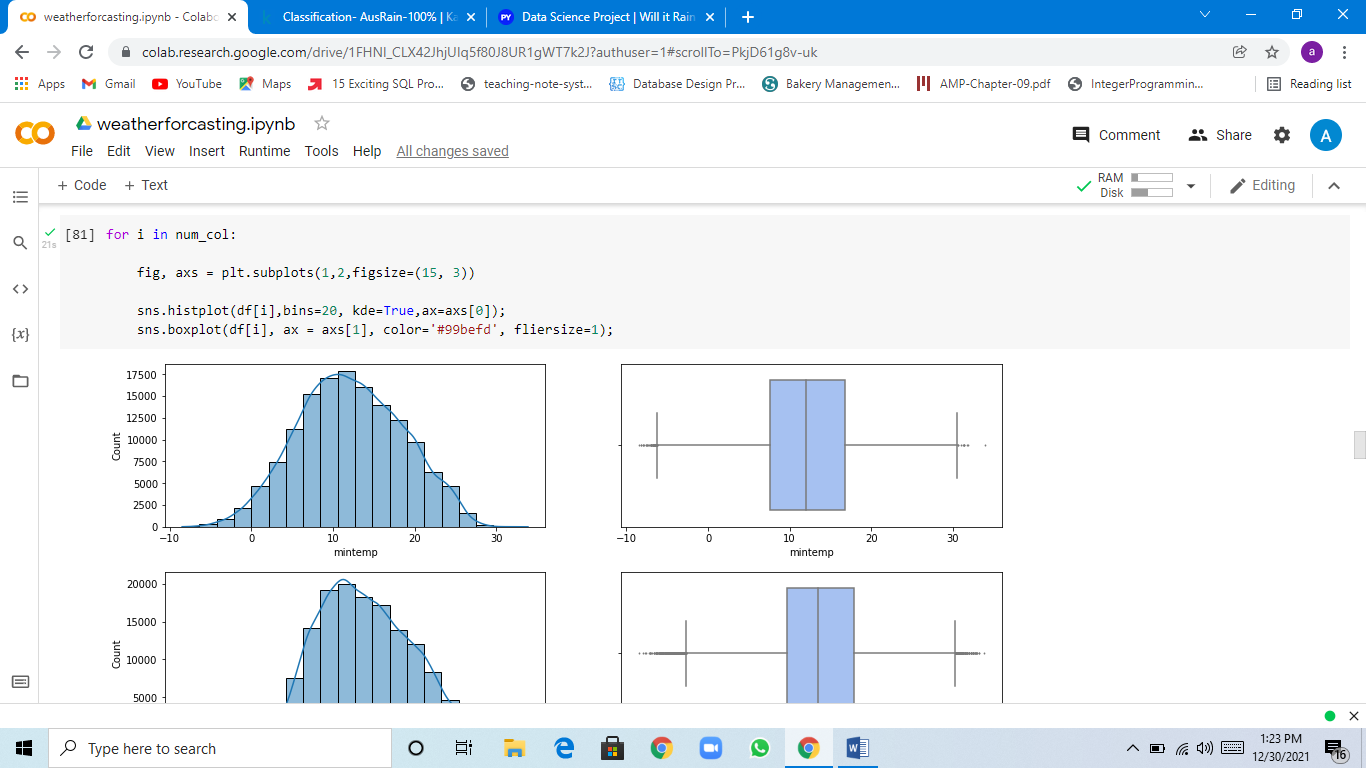
* we know that there are some different number of missing values to each features
* Depending on their distribution we are going to replace with median or mean

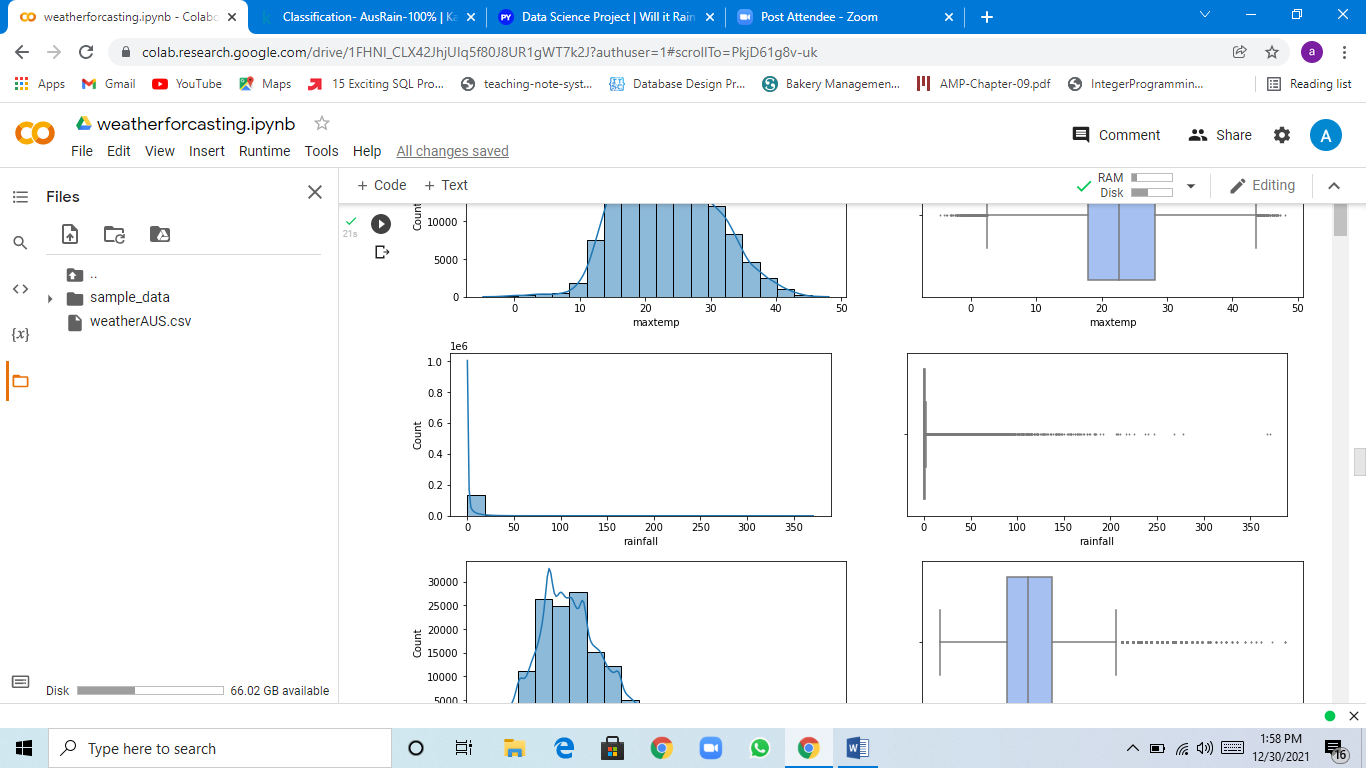
for i in num\_col:

fig, axs = plt.subplots(1,2,figsize=(15, 3))

 sns.histplot(df[i],bins=20, kde=True,ax=axs[0]);

 sns.boxplot(df[i], ax = axs[1], color='#99befd', fliersize=1);



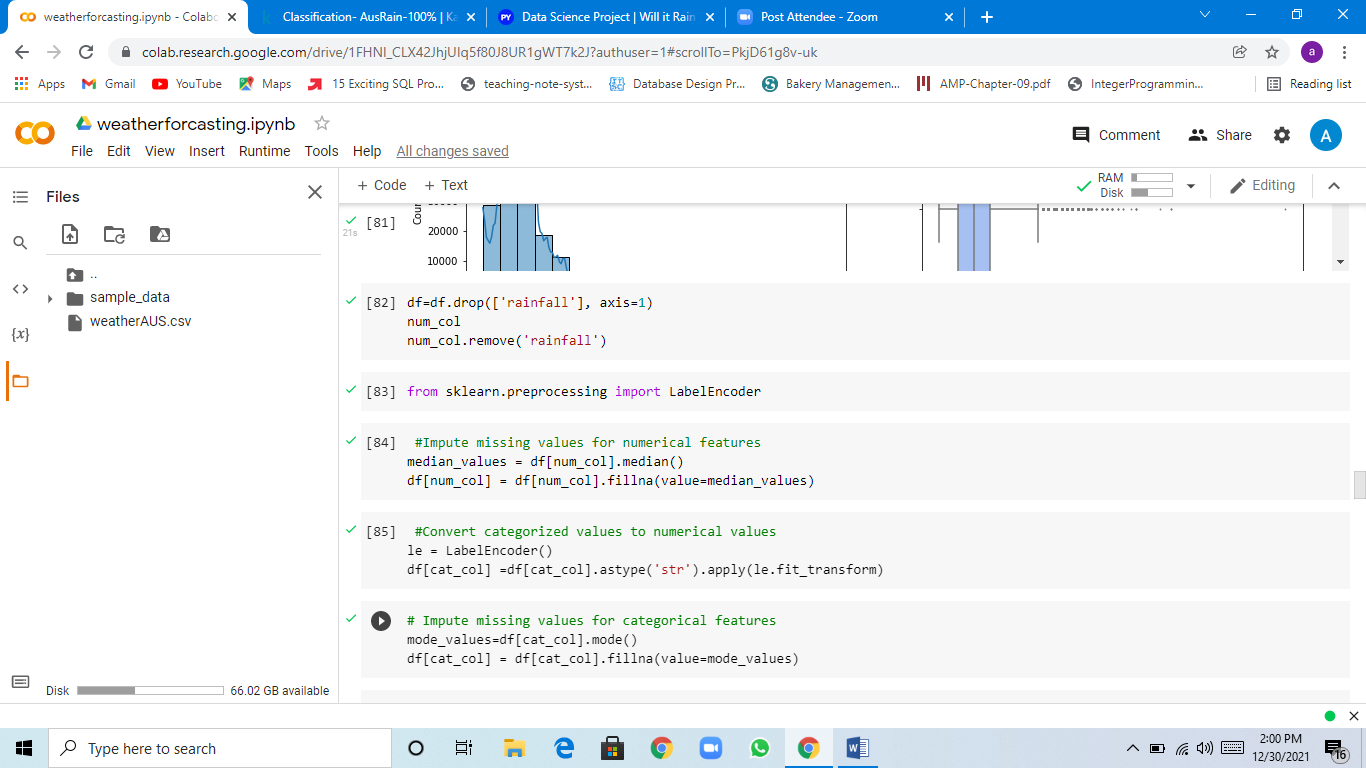


This graph shows that the rainfall feature seems to have huge distortion,we will remove this feature for our data analysis.

df=df.drop(['rainfall'], axis=1)

num\_col

num\_col.remove('rainfall')



**4-PRE-PROCESSING:**

Before modeling we have to do two steps on our data,endocing and dealing with missing values.

We do encoding because we have categorical data.

Encoding:

It is a process by which categorical data converted into numerical for processing.

from sklearn.preprocessing import LabelEncoder

 #Impute missing values for numerical features

median\_values = df[num\_col].median()

df[num\_col] = df[num\_col].fillna(value=median\_values)

 #Convert categorized values to numerical values

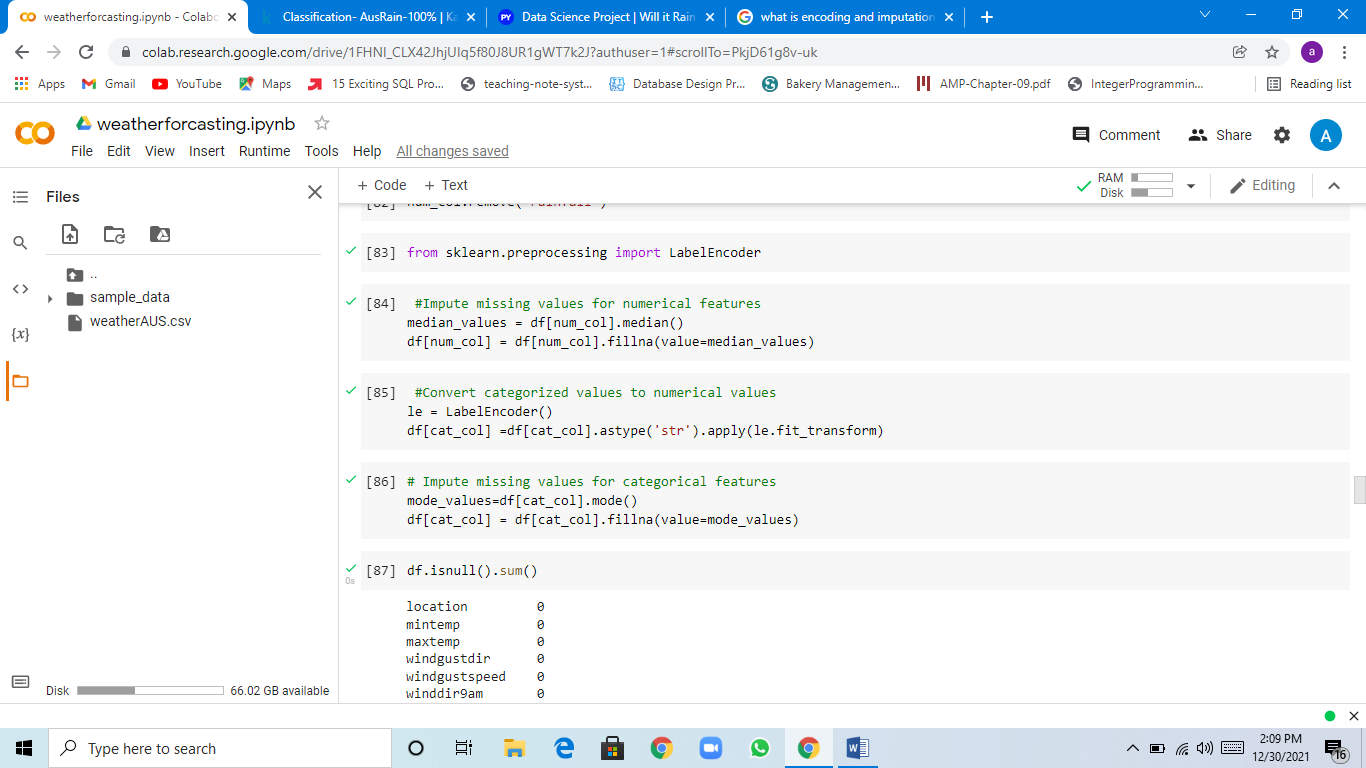
le = LabelEncoder()

df[cat\_col] =df[cat\_col].astype('str').apply(le.fit\_transform)

# Impute missing values for categorical features

mode\_values=df[cat\_col].mode()

df[cat\_col] = df[cat\_col].fillna(value=mode\_values)



CORRELATION:

Correlation means association - more precisely it is a measure of the extent to which two variables are related. There are three possible results of a correlational study: a positive correlation, a negative correlation, and no correlation.

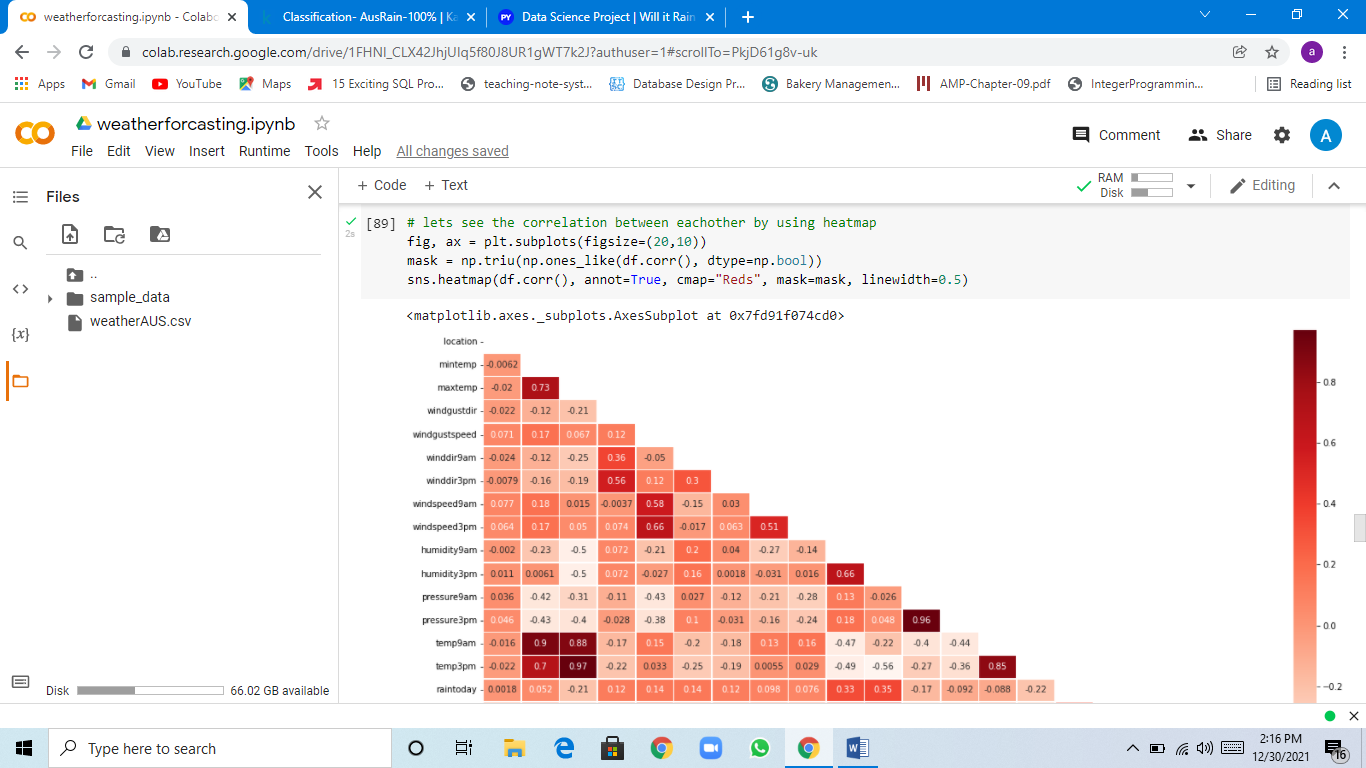
* POSITIVE CORRELATION:directly relation between two variables..
* NEGATIVE CORRELATION: inversely relation between two variables.
* ZERO CORRELATION: there is no relation between two variables.

# lets see the correlation between eachother by using heatmap

fig, ax = plt.subplots(figsize=(20,10))

mask = np.triu(np.ones\_like(df.corr(), dtype=np.bool))

sns.heatmap(df.corr(), annot=True, cmap="Reds", mask=mask, linewidth=0.5)



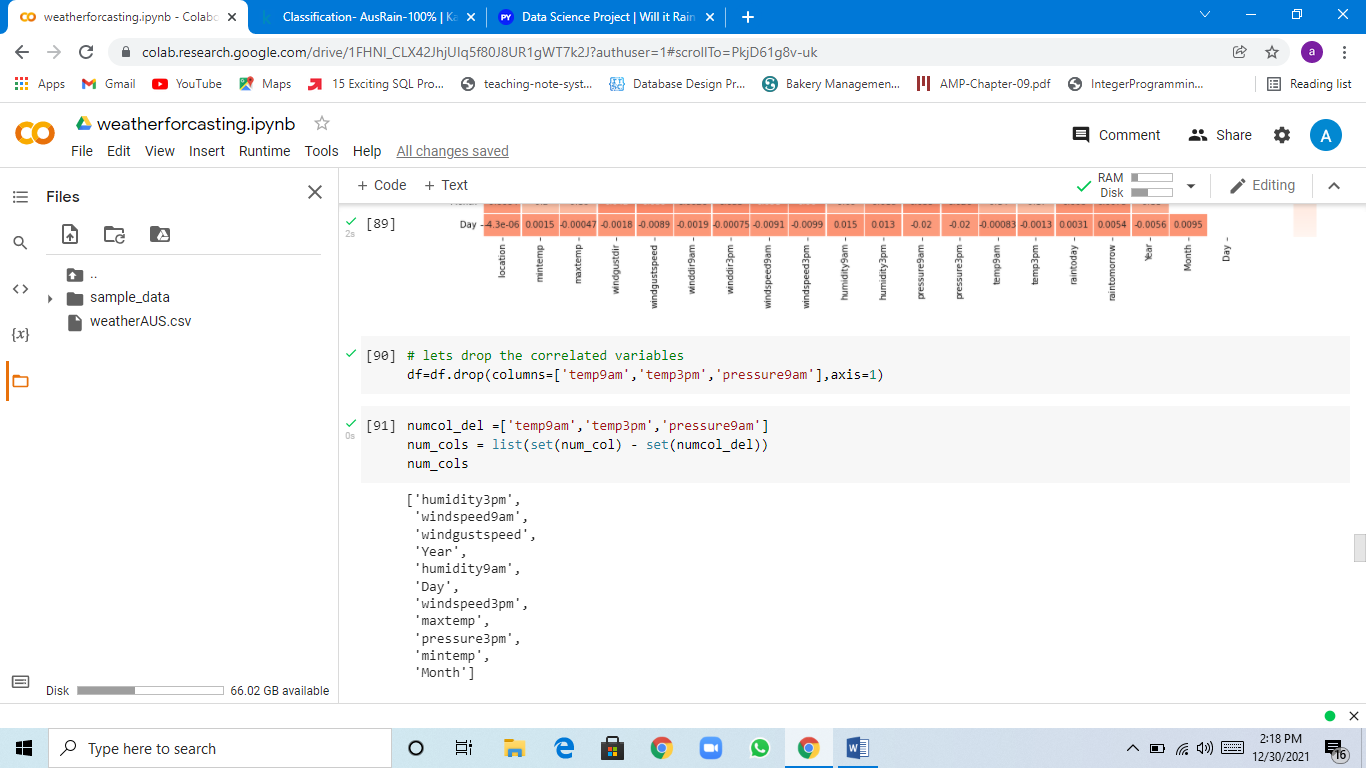
# lets drop the correlated variables

df=df.drop(columns=['temp9am','temp3pm','pressure9am'],axis=1)

numcol\_del =['temp9am','temp3pm','pressure9am']

num\_cols = list(set(num\_col) - set(numcol\_del))

num\_cols

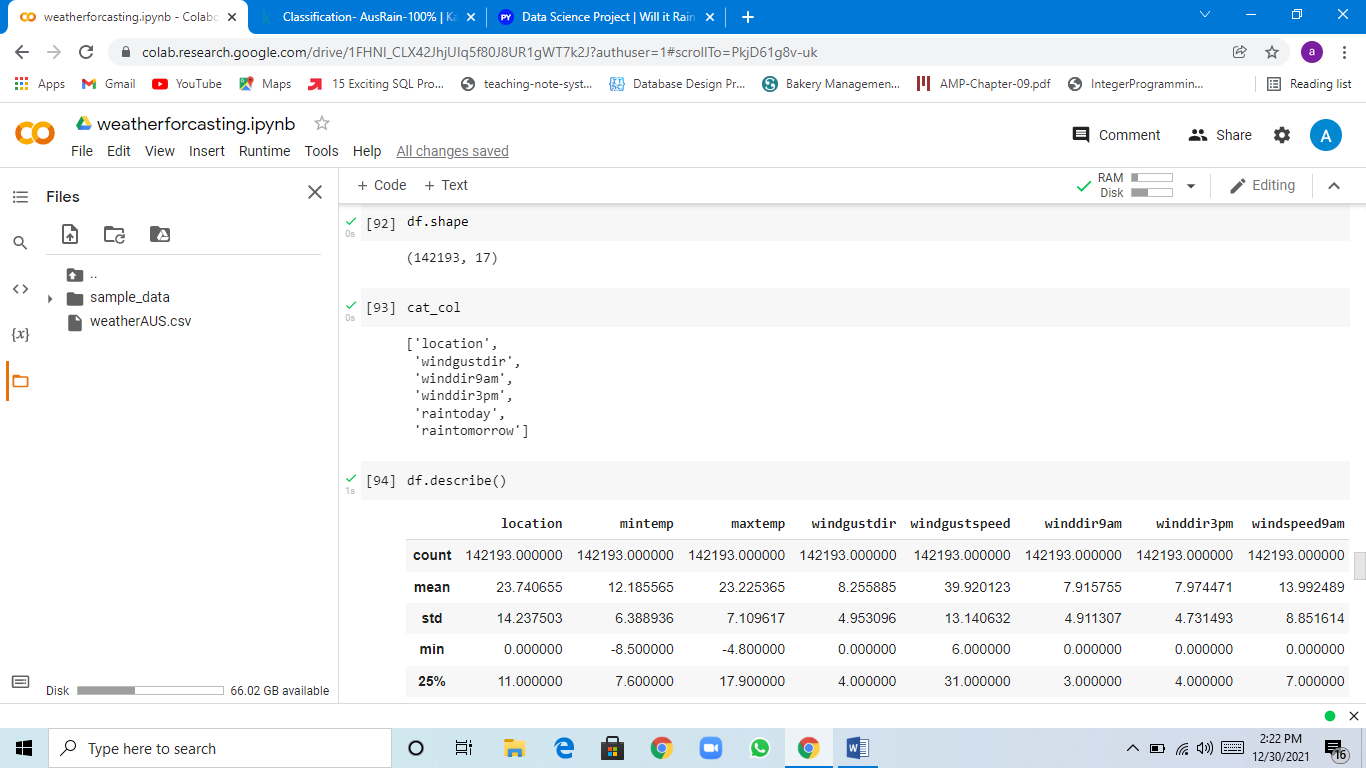


**5-DATASCALING:**

df.shape

cat\_col

df.describe()



THE DATA IS READY FOR MODELING.

First we split the data into train and test data.

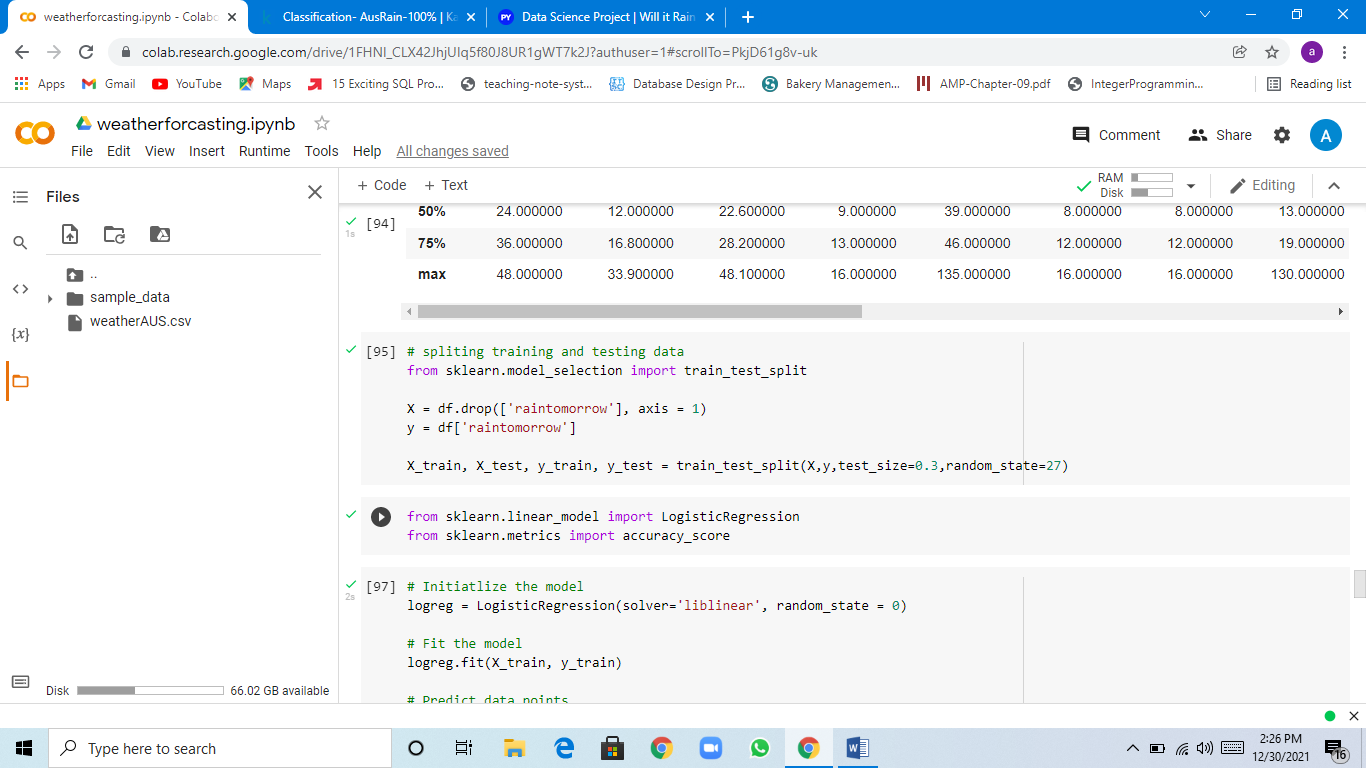
# spliting training and testing data

from sklearn.model\_selection import train\_test\_split

X = df.drop(['raintomorrow'], axis = 1)

y = df['raintomorrow']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=0.3,random\_state=27)



MODELS:

1-LOGISTIC REGRESSION:

 Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

# Initiatlize the model

logreg = LogisticRegression(solver='liblinear', random\_state = 0)

# Fit the model

logreg.fit(X\_train, y\_train)

# Predict data points

y\_pred\_test = logreg.predict(X\_test)

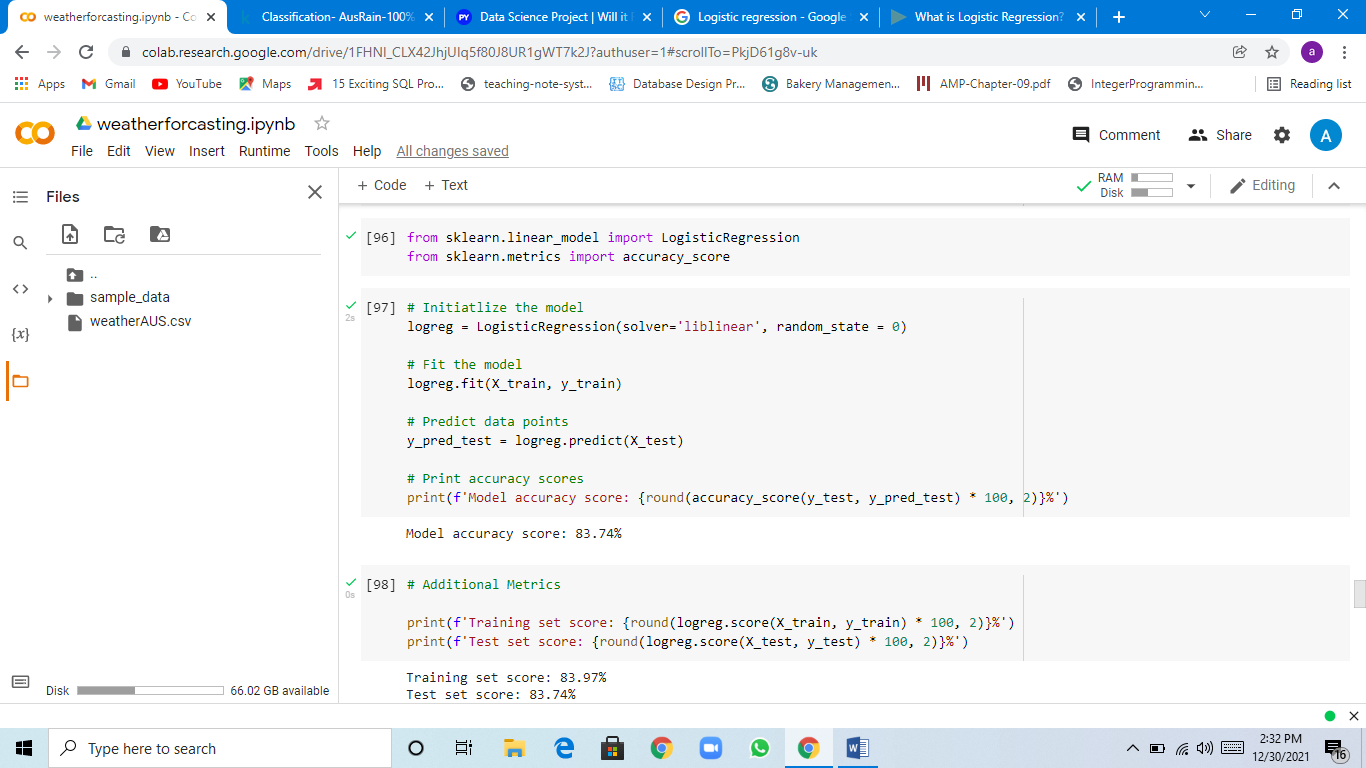
# Print accuracy scores

print(f'Model accuracy score: {round(accuracy\_score(y\_test, y\_pred\_test) \* 100, 2)}%')

# Additional Metrics

print(f'Training set score: {round(logreg.score(X\_train, y\_train) \* 100, 2)}%')

print(f'Test set score: {round(logreg.score(X\_test, y\_test) \* 100, 2)}%')



2-RANDOM FOREST:

Random forest is an ensemble technique which is better than decision tree. It always give low variance.

One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems.



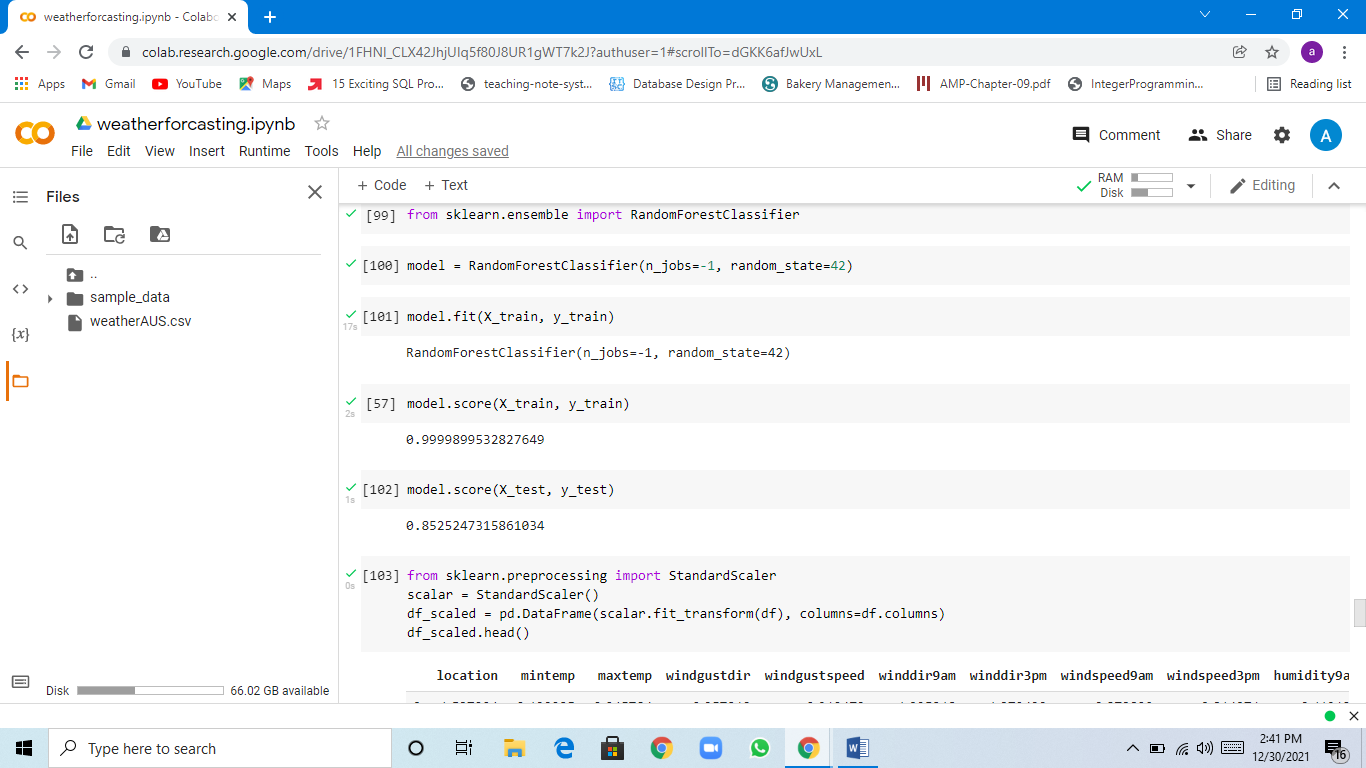
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n\_jobs=-1, random\_state=42)

model.fit(X\_train, y\_train)

model.score(X\_train, y\_train)

model.score(X\_test, y\_test)



**3-PCA:**

 This method combines highly correlated variables together to form a smaller number of an artificial set of variables which is called “principal components” that account for most variance in the data.

STEPS:

1. Standardization
2. Covariance matrix computation
3. Compute the eigen vectors and eigen values

from sklearn.preprocessing import StandardScaler

scalar = StandardScaler()

df\_scaled = pd.DataFrame(scalar.fit\_transform(df), columns=df.columns)

df\_scaled.head()

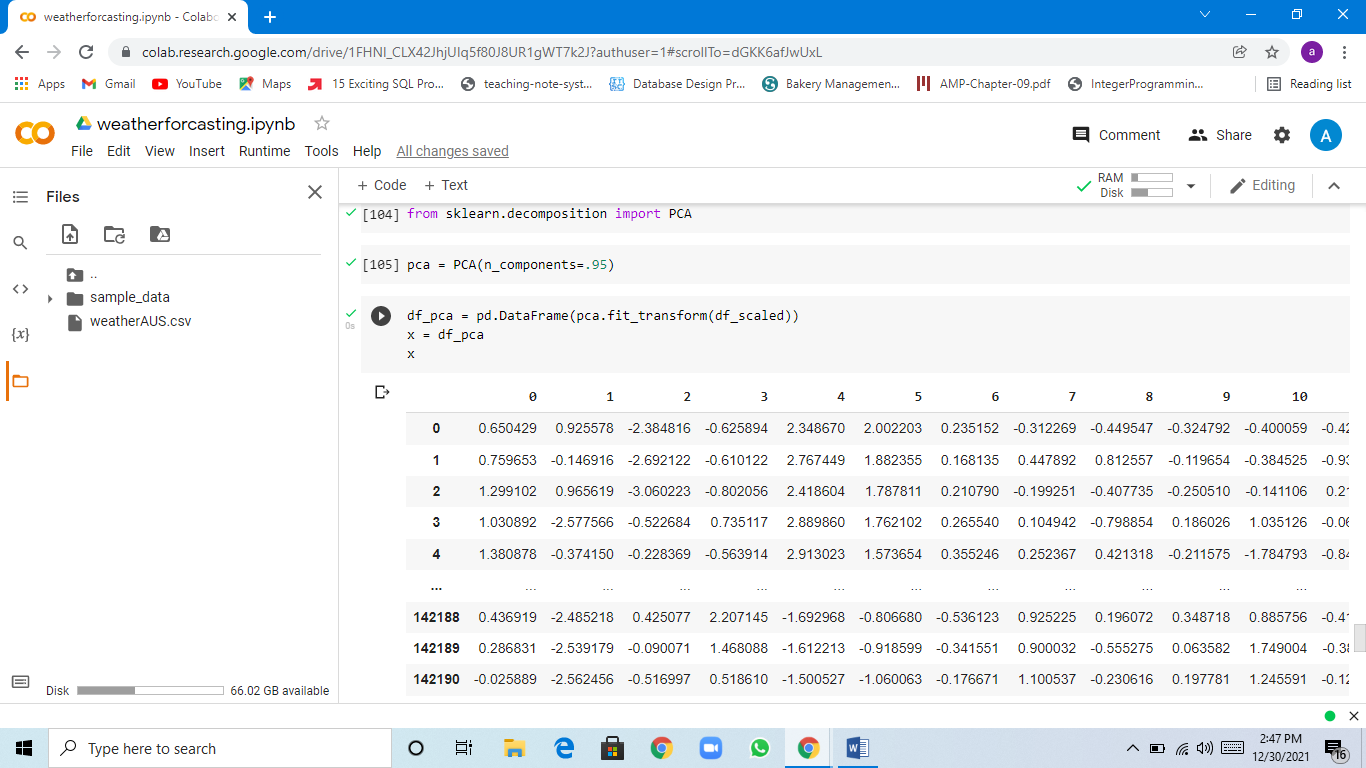
from sklearn.decomposition import PCA

pca = PCA(n\_components=.95)

df\_pca = pd.DataFrame(pca.fit\_transform(df\_scaled))

x = df\_pca

x



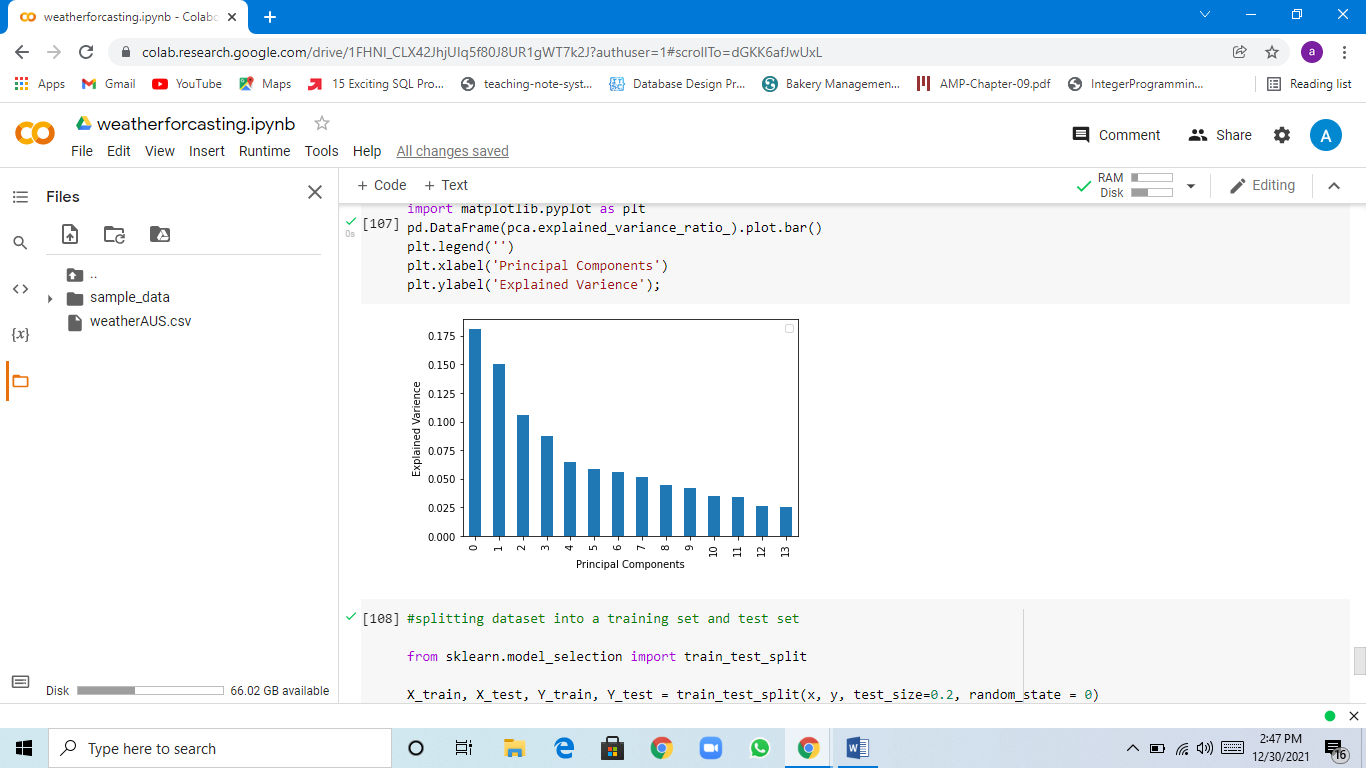
import matplotlib.pyplot as plt

pd.DataFrame(pca.explained\_variance\_ratio\_).plot.bar()

plt.legend('')

plt.xlabel('Principal Components')

plt.ylabel('Explained Varience');



#splitting dataset into a training set and test set

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, random\_state =

#fitting logistic Regression to training set

from sklearn.linear\_model import LogisticRegression

classifier = LogisticRegression(random\_state = 0)

classifier.fit(X\_train, Y\_train)

y\_pred = classifier.predict(X\_test)

print("accuracy score:", accuracy\_score(Y\_test,y\_pred))

