SIT742 Modern Data Science

Assignment 2

Team Members:

Student Name	Muskan Mangla	Parmeet Kaur	Vijender Beniwal
Student ID	218557767	219248489	218650365

Date: 22 May 2021

Table of Contents

Part I - Data Analytic — Web Log Data

Hotel TULIP a five-star hotel located at Deakin University, and its CIO Dr Bear Guts has asked the Team-SIT742 team to analyse the weblogs files. As an employee for Hotel Tulip, working in the Information Technology Division, it is required to prepare a set of documentation for Team-SIT742 to allow them to understand the data being dealt with. Throughout this report, some source codes are to explore the weblog, which afterwards the information is presented to Dr Bear Guts in the format of a report.

1. Data ETL

1.1. Load Data

Load data and read data into finaldf (dataframe)

```
from ipywidgets import IntProgress
import pandas as pd
from IPython.display import display
from zipfile import ZipFile
import pandas as pd
import os


# # Your file might be in a different location, so you need to customize the path
z = ZipFile('/content/drive/My Drive/HTWebLog_p1.zip','r')
dfs=[]
for file in z.namelist():
    dfs.append(pd.read_csv(z.open(file), sep=' ',header=None,encoding="ISO-8859-1", quotechar='"',comment='#',error_bad_lines=False))
    finaldf = pd.concat(dfs)
    print(finaldf)
```

remove all NAs

For the Columns if the column is with 15% NA'S, Then need to remove that column

```
[18] percent_15missing_columndrop = finaldf.loc[:, finaldf.isnull().mean() < .15]
```

For the rows, if there any contains Na' in row then remove the Row

```
[19] #Your code to remove missing values as required.
     df_ht = percent_15missing_columndrop.dropna()
```

randomly select 30% of the total data

```
# only 30% of total data are selected for classification
weblog_df = df_ht.sample(frac = 0.3, random_state=1)
```

A. The number of requests in weblog_df.
Number of request before removing missing value

```
print("requests before NA's remove",finaldf.shape[0])
```

```
requests before NA's remove 8224960
```

Number of requests after removing missing value and after randomly selection 30% of total data.

```
#Your code to show the number of requests in weblog_df
weblog_df.count()
print("number of requests in weblog_df:",weblog_df.shape[0])
```

```
number of requests in weblog_df: 2467261
```

1.2. Feature Selection

Select 'cs_method', 'cs_ip', 'cs_uri_stem', 'cs(Referer)' as input features and 'sc_status' as class label into a new dataframe ml_df .

```
# Your code for feature selection and label selection
ml_df = weblog_df[['cs-method', 'c-ip','cs-uri-stem', 'cs(User-Agent)','sc-status']]
```

A. Data Description of ml_df.

```
ml_df.describe()
```

	sc-status
count	2.467261e+06
mean	2.301088e+02
std	5.082895e+01
min	2.000000e+02
25%	2.000000e+02
50%	2.000000e+02
75%	3.040000e+02
max	5.010000e+02

We can conclude that the total count number of non-NA observation are 2.467261e+06and minimum value for ‘sc-status’ is 2.00000e+02 whereas minimum observation in the object is 5.010000e+02. The datatype of features are object and label is float64.

```
ml_df.dtypes
```

```
cs-method          object
c-ip               object
cs-uri-stem        object
cs(User-Agent)     object
sc-status         float64
dtype: object
```

B. Top 5 rows of ml_df.

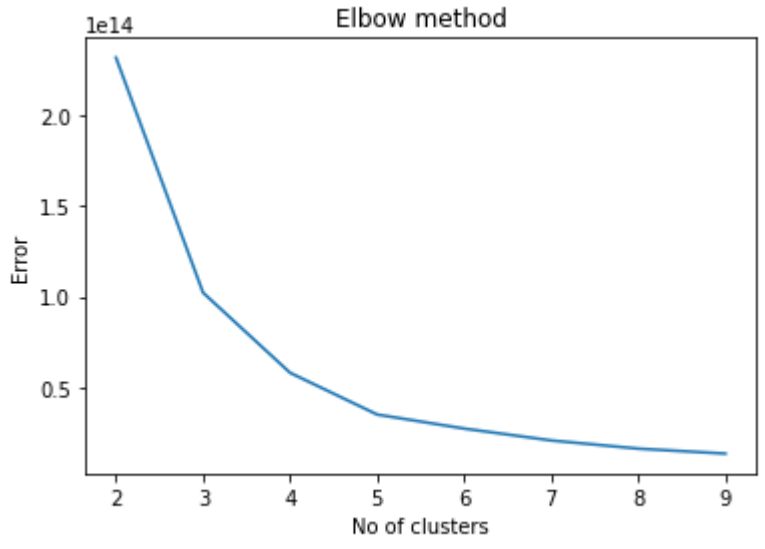
```
#Your code to show the top 5 rows of ml_df
ml_df.head()
```

	cs-method	c-ip	cs-uri-stem	cs(User-Agent)	sc-status
14999	GET	202.126.216.5	/Tulip/common/zh-hk/images/sidebanner_2.jpg	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	200.0
14322	GET	218.103.20.6	/Tulip/common/zh-hk/images/sidebanner_1.jpg	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	200.0
27231	GET	121.35.50.185	/Tulip/common/en-us/images/sidebanner_11.jpg	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	304.0
15125	GET	219.78.68.194	/Tulip/common/common_style.aspx	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	200.0
5991	GET	203.198.226.102	/promotion/FP00207_2.jpg	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	200.0

2. Unsupervised learning

Perform unsupervised learning on ml_df with K Means, with a varying K from 2 to 10 only using #sklearn.

A. Figure 'KMeans' in the elbow plot, with a varying K from 2 to 10.



B. Best K for dataset

the

3. Supervised learning

3.1. Data Preparation

For data preparing , we are going to install and import the useful package an libraries such as PySpark and sparksession from pyspark.sql. after that we are going to create schema of features

```
schema = StructType([StructField("sc_status", IntegerType(), True),
                     StructField("cs_method", IntegerType(), True),
                     StructField("c_ip", IntegerType(), True),
                     StructField("cs_uri_stem", IntegerType(), True),
                     StructField("cs(User_Agent)", IntegerType(), True)])
```

and label and using spark, 'creating sl_df'

```
sl_df = spark.createDataFrame(le_df, schema)
```

as well as only using 10% of data is going to use to next steps.

```
#Only 10% of the data is used in this part.
sl_df = sl_df.sample(fraction=0.1, seed=1)
```

After we implement vector assembler which is useful for combining raw features and features genitaled by different feature transformed into a sector feature vector

```
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
# transformer
vector_assembler = VectorAssembler(inputCols=['cs_method', 'c_ip', 'cs_uri_stem', 'cs(User_Agent)'],outputCol="features")
df_temp = vector_assembler.transform(sl_df)
df_temp.show(3)
```

```
+---------+---------+----+-----------+--------------+--------------------+
|sc_status|cs_method|c_ip|cs_uri_stem|cs(User_Agent)|            features|
+---------+---------+----+-----------+--------------+--------------------+
|        0|    41957|1298|       1382|             0|[41957.0,1298.0,1...|
|        0|    20918|1873|       1525|             4|[20918.0,1873.0,1...|
|        0|    56286|1567|       1525|             4|[56286.0,1567.0,1...|
+---------+---------+----+-----------+--------------+--------------------+
only showing top 3 rows
```

3.2. Logistic Regression

Perform supervised learning on dataset with Logistic Regression only using #PySpark

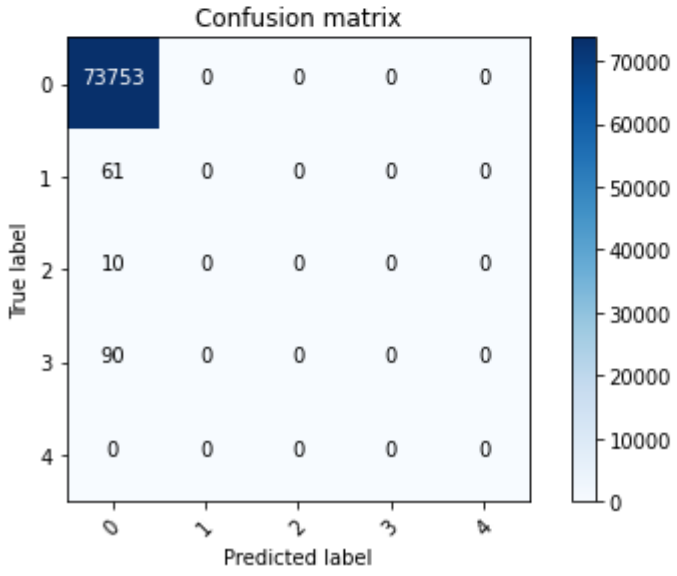
```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
# Your code contains trainning from train data and predicting based on the test data
lgr = LogisticRegression(featuresCol = 'features', labelCol = 'sc_status', maxIter=5)
lgrModel = lgr.fit(trainingData)
model_predict = lgrModel.transform(testData)
model_predict.select('sc_status', 'features',  'rawPrediction', 'prediction', 'probability').toPandas().head(5)
```

	sc_status	features	rawPrediction	prediction	probability
0	0	[2.0, 1361.0, 1525.0, 0.0]	[6.932932438143316, -0.01932303187936868, -3.1...	0.0	[0.9979217329826513, 0.0009544876060551948, 4....
1	0	[3.0, 1349.0, 866.0, 0.0]	[6.933506598448185, -0.23055271444259906, -3.0...	0.0	[0.9979574062798311, 0.0007723258623183917, 4....
2	0	[4.0, 1104.0, 901.0, 0.0]	[6.929822116584948, -0.21480643280153633, -3.0...	0.0	[0.9980426935729347, 0.0007875467894609735, 4....
3	0	[10.0, 1359.0, 1382.0, 0.0]	[6.933147465676945, -0.06518097753553675, -3.1...	0.0	[0.9979344107265146, 0.0009115207779593333, 4....
4	0	[11.0, 1560.0, 4185.0, 0.0]	[6.932997733215159, 0.8304939670098502, -3.411...	0.0	[0.9970502945420163, 0.0022306598718597145, 3....

A. Display the classification result using confusion matrix including TP, TN, FP, FN,

```
from sklearn.metrics import confusion_matrix
import numpy as np

# Your code to display TP, TN, FP, FN

y_true = model_predict.select("sc_status")
y_true = y_true.toPandas()

y_pred = model_predict.select("prediction")
y_pred = y_pred.toPandas()

cnf_matrix = confusion_matrix(y_true, y_pred,labels=class_names)
#cnf_matrix
plt.figure()
confusion_matrix_plot(cnf_matrix, classes=class_names,
                      title='Confusion matrix')
plt.show()
```

```
Confusion matrix, without normalization
[[73753     0     0     0     0]
 [   61     0     0     0     0]
 [   10     0     0     0     0]
 [   90     0     0     0     0]
 [    0     0     0     0     0]]
```



True negative is the actual value was false as model was predicted false where false positive means that actual value was false but model predicted true.

Next, false negative states that actual value was true and but model precited false.

Lastly, true positive shows the actual value was true as model predicted it as true.

B. *Display the classification result using Precision, Recall and F1 score.*

```
from sklearn.metrics import classification_report

# Your Code to display the classification results as required.
import sklearn.metrics as skm
cm = skm.multilabel_confusion_matrix(y_true, y_pred)
print("Classification Report:", skm.classification_report(y_true,y_pred))
```

```
Classification Report:              precision    recall  f1-score   support

           0       1.00      1.00      1.00     73753
           1       0.00      0.00      0.00        61
           2       0.00      0.00      0.00        10
           3       0.00      0.00      0.00        90
           5       0.00      0.00      0.00         1

    accuracy                           1.00     73915
   macro avg       0.20      0.20      0.20     73915
weighted avg       1.00      1.00      1.00     73915

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: U
  _warn_prf(average, modifier, msg_start, len(result))
```

3.3. K-fold Cross Validation

A. *Your code design and running results,*

First model is Decision tree classification with multiclassclassififactionevaluator.

```
from pyspark.ml import Pipeline
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator

# K = 2
# Your code for 2-fold cross validation
rf = DecisionTreeClassifier(featuresCol = 'features', labelCol = 'sc_status')

rfevaluator = MulticlassClassificationEvaluator(predictionCol="prediction", labelCol="sc_status")

rfparamGrid = (ParamGridBuilder()
             #.addGrid(rf.maxDepth, [2, 5, 10, 20, 30])
               .addGrid(rf.maxDepth, [2, 5, 10])
             #.addGrid(rf.maxBins, [10, 20, 40, 80, 100])
               .addGrid(rf.maxBins, [5, 10, 20])
             #.addGrid(rf.numTrees, [5, 20, 50, 100, 500])

             .build())

# Create 2-fold CrossValidator
rfcv = CrossValidator(estimator = rf,
                      estimatorParamMaps = rfparamGrid,
                      evaluator = rfevaluator,
                      numFolds = 2)

# Run cross validations.
rfcvModel = rfcv.fit(trainingData)
print(rfcvModel)

# Use test set here so we can measure the accuracy of our model on new data
rfpredictions = rfcvModel.transform(testData)

# cvModel uses the best model found from the Cross Validation
# Evaluate best model
print('RMSE:', rfevaluator.evaluate(rfpredictions))
```

```
CrossValidatorModel_77dbd44750ab
RMSE: 0.9984306949602614
```

Second one is linera regression

```
from pyspark.ml.regression import LinearRegression
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
from pyspark.ml.evaluation import RegressionEvaluator

# Create initial LinearRegression model
lr = LinearRegression(labelCol="sc_status", featuresCol="features")


# Create ParamGrid for Cross Validation
lrparamGrid = (ParamGridBuilder()
             .addGrid(lr.regParam, [0.5, 1.0, 2.0])
             #  .addGrid(lr.regParam, [0.01, 0.1, 0.5])
             .addGrid(lr.elasticNetParam, [0.5, 0.75, 1.0])
             #  .addGrid(lr.elasticNetParam, [0.0, 0.5, 1.0])
             .addGrid(lr.maxIter, [1, 5, 10,])
             #  .addGrid(lr.maxIter, [1, 5, 10])
             .build())

# Evaluate model
lrevaluator = RegressionEvaluator(predictionCol="prediction", labelCol="sc_status", metricName="rmse")

# Create 5-fold CrossValidator
lrcv = CrossValidator(estimator = lr,
                    estimatorParamMaps = lrparamGrid,
                    evaluator = lrevaluator,
                    numFolds = 2)

# Run cross validations
lrcvModel = lrcv.fit(trainingData)
print(lrcvModel)

# Get Model Summary Statistics
lrcvSummary = lrcvModel.bestModel.summary

# Use test set here so we can measure the accuracy of our model on new data
lrpredictions = lrcvModel.transform(testData)

# cvModel uses the best model found from the Cross Validation
# Evaluate best model
print('RMSE:', lrevaluator.evaluate(lrpredictions))
```

```
CrossValidatorModel_2b53231b54b6
RMSE: 0.11242858259347613
```

Third one is Naïve bayes

```
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
from pyspark.ml.evaluation import BinaryClassificationEvaluator,MulticlassClassificationEvaluator

# Create initial Naïve Bayes model
nb = NaiveBayes(labelCol="sc_status", featuresCol="features")

# Create ParamGrid for Cross Validation
nbparamGrid = (ParamGridBuilder()
               .addGrid(nb.smoothing, [0.0, 0.2, 0.4, 0.6, 0.8, 1.0])
               .build())

# Evaluate model
nbevaluator = MulticlassClassificationEvaluator(predictionCol="prediction", labelCol="sc_status")

# Create 2-fold CrossValidator
nbcv = CrossValidator(estimator = nb,
                      estimatorParamMaps = nbparamGrid,
                      evaluator = nbevaluator,
                      numFolds = 2)

# Run cross validations
nbcvModel = nbcv.fit(trainingData)
print(nbcvModel)

# Use test set here so we can measure the accuracy of our model on new data
nbpredictions = nbcvModel.transform(testData)

# cvModel uses the best model found from the Cross Validation
# Evaluate best model
print('RMSE:', nbevaluator.evaluate(nbpredictions))
```

```
CrossValidatorModel_1ec0bb199ebd
RMSE: 0.6345646536695481
```

B. *Your findings on hyper-parameters based on this cross-validation results (Best results).*

After comparing the RMSE value for above three model we find the which model has less RMSE that model is good at predicting the observed data. because low RMSE means low noise in data. We make perfect to observe the data. If we have large RMSE, which indicates that our model is failing to account for useful feature of data underlying our data. So conclude that linear regression model is best model at predicting the observed data as compare to other twos.

4. Association Rule Mining

A. *Your code design and running results, threshold (parameter) for support and confidence*

```
# you can also use PySpark package, if preferred
from apyori import apriori

# Your code
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('seaborn-deep')

ds = ml_df.values

# Creating the transactions list of lists
transactions = []

for i in range(ds.shape[0]):
    transactions.append([str(ds[i][j]) for j in range(ds.shape[1])])

# Training Apriori on the dataset
rules = apriori(transactions, min_support = 0.003, min_confidence = 0.2,
                min_lift = 5, min_length = 2)

# Visualising the results
results = list(rules)
results
```

```
[RelationRecord(items=frozenset({'404.0', '/favicon.ico'}), support=0.008888804224603721, ordered_statistics=[OrderedStatistic(
 RelationRecord(items=frozenset({'404.0', '/favicon.ico', 'GET'}), support=0.008887993609107428, ordered_statistics=[OrderedSta
```

B. *Your findings on association rule mining results*

```
ass_results = list(results)

for ass_item in ass_results:
    pair = ass_item[0]
    pair_item = [x for x in pair]
    print("Rule: " + pair_item[0] + " ==> " + pair_item[1])

    #second index of the inner list
    print("Support: " + str(ass_item[1]))
    print("Confidence: " + str(ass_item[2][0][2]))
    print("\n")
```

```
Rule: 404.0 ==> /favicon.ico
Support: 0.008888804224603721
Confidence: 1.0


Rule: 404.0 ==> /favicon.ico
Support: 0.008887993609107428
Confidence: 0.999908804888058
```

the support represents the popularity of a special product of all product transaction and it is calculated by the ratio of number of transaction includes that product and total number of transaction. According to rule: we calculate the support is 0.8888880422460371 and confidence with first item is 1.0 and confidence with another pair item is 0.99. confidence can be described as likelihood of purchasing both products A and B. So in our scenario likelihood of error 404.0 due to favicon.ico is 1.0.

Part II - Web Crawling

In 2021, to better introduce and understand the research works on the professors, Deakin university wants to perform the citation prediction on individual professor level. You are required to implement a web crawler to crawl the citation information for Gang Li from 2003 to 2021 (start at 2003 and end at 2021), and also conduct several prediction coding tasks. You will need to make sure that the web crawling code and prediction code meets the requirements. You are free to use any Python package for Web crawling and prediction by finishing below tasks.

5. Crawl the Gang Li citation information from 2003 to 2021

My code to generate the csv for Gang Li's citation from 2003 to 2021

a. Install required library scholarly to crawl data

```
# required lib to crawl data from url
!pip install scholarly
```

b. Code to crawl the citations of Professor Gang Li's google scholar profile page from year 2003 to 2021

```
from scholarly import scholarly

search_data = scholarly.search_author('Gang Li, Deakin University')
get_author = next(search_data)
get_author = scholarly.fill(get_author, sections=['counts'])
create_df = pd.DataFrame([get_author['cites_per_year'].keys(),get_author['cites_per_year'].values()]).T

create_df.columns=["Year","Citation"]
create_df.sort_values("Year",inplace=True)
create_df.set_index("Year",inplace=True)

# save info to csv
create_df.to_csv('citation2003-2021.csv')
```

c. Data extraction results:

```
create_df
```

Year	Citation
2003	15
2004	34
2005	17
2006	11
2007	33
2008	41
2009	57
2010	68
2011	105
2012	131
2013	170
2014	251
2015	290
2016	340
2017	385
2018	452
2019	583
2020	842
2021	322

6. Train Arima to predict the 2018 to 2020 citation

6.1. Train Arima Model

Use the create_df, and perform the Arima training with parameter of p=1, q=1 and d=1 on data from 2003 to 2017 (15 years)

```
!pip install --upgrade --no-deps statsmodels
from statsmodels.tsa.arima.model import ARIMA
```

```
Requirement already up-to-date: statsmodels in /usr/local/lib/python3.7/dist-packages (0.12.2)
```

```
# your code to use create_df to split the data into train (year 2003 to 2017) and test
train=create_df.iloc[create_df.index<2018]
test=create_df.iloc[(create_df.index>=2018)&(create_df.index<=2020)]

model = ARIMA(train, order=(1,1,1))
result = model.fit()
```

6.2. Predicting the citation and Calculate the RMSE

Predict the citation on year 2018, 2019 and 2020

```
# your code to predict the citation and save it to variable pred
preds=result.predict(start=2018,end=2020)#,dynamic=True)
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/base/tsa_
  ValueWarning)
```

```
print(preds)
print(test)
```

```
2018    1198.45466
2019    1198.45466
2020    1198.45466
```

perform the evaluation by comparing the predicted citation from 2018 to 2020 with the true citation from 2018 to 2020

```
print(preds)
print(test)
```

```
2018    1198.45466
2019    1198.45466
2020    1198.45466
Name: predicted_mean, dtype: float64
      Citation
Year
2018       452
2019       583
2020       842
```

and calculate the RMSE

```
# Print the error below by comparing the test and preds:
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import numpy as np

# Your code to show the performance RMSE
print("RMSE (2018 to 2020):",mean_squared_error(np.array(test),np.array(preds))**0.5)
```

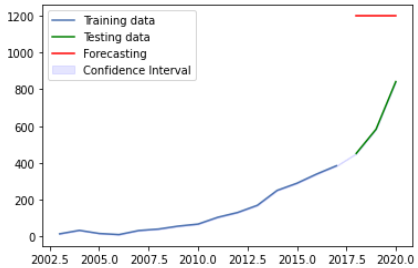
```
RMSE (2018 to 2020): 595.2699449319601
```

6.3. Draw the visualization to compare

Use matplotlib to draw the line plot with training data from 2013 to 2017, the testing truth, the prediction and also the confidence interval (95%).

```
# You code: Visualize as required, the prediction with its confidence interval
ci = 1.96 * np.std(create_df.iloc[:-1])/np.mean(create_df.iloc[:-1])

plt.plot(train)
plt.plot(test,c="Green")
plt.plot(preds,c="Red")
plt.fill_between(create_df.iloc[:-1].index, np.squeeze(create_df.iloc[:-1]-ci), np.squeeze(create_df.iloc[:-1]+ci), color='b', alpha=.1)
plt.legend(["Training data","Testing data","Forecasting","Confidence Interval"])
```

```
<matplotlib.legend.Legend at 0x7f70e1e80a90>
```



7. Parameter selection and Year 2021 and 2022 Prediction

7.1. Grid Search

Run the grid search for parameter p=[1,2], q=[1,2], d=[1,2] with training data on year 2003 to 2017 and testing data on 2018 to 2020, and save results of the search on each parameter combination in the "search-results.csv"

```
p = [1,2]
q = [1,2]
d = [1,2]
PARAMETER=[]
RMSE=[]

resuls_grid=pd.DataFrame(columns=["PARAMETER","RMSE"])
def evaluate_models(train_dataset,p_values,d_values,q_values):
    for p in p_values:
        for d in d_values:
            for q in q_values:
                model = ARIMA(train_dataset['Citation'], order=(p,d,q))
                result = model.fit()
                prediction=result.predict(start=2018,end=2020,dynamic=True)
                RMSE_value=(mean_squared_error(test['Citation'],prediction))**0.5
                PARAMETER.append((p,d,q))
                RMSE.append(RMSE_value)

evaluate_models(train,p,d,q)
```

Results of seach-results.csv with RMSE and parameter set column

```
# your code to generate the seach-results.csv and print the top 6 rows

Results = pd.DataFrame({'RMSE':RMSE,'Parameter':PARAMETER})
Results.to_csv("seach-results.csv")
Results
```

	RMSE	Parameter
0	595.269945	(1, 1, 1)
1	317.867876	(1, 1, 2)
2	94176.050701	(1, 2, 1)
3	95620.864925	(1, 2, 2)
4	574.772954	(2, 1, 1)
5	183.209271	(2, 1, 2)
6	82671.019253	(2, 2, 1)
7	72882.007424	(2, 2, 2)

7.2. Select the best parameter values and Predict for 2021 and 2022

Please add descriptions of the following contents by yourselves.

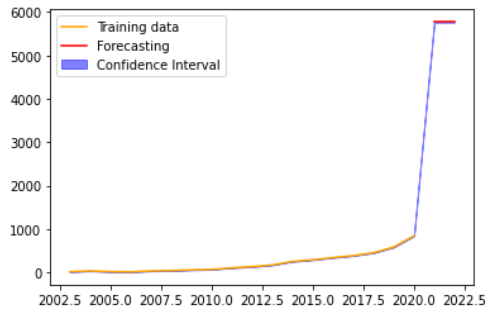
A. *Find out and display the best parameter values*
After seeing the output of search-results.csv we can conclude that best parameter is with least RMSE value which it is best model to predict the data and it has least noise and outlier. So parameter: 2,1,2 is best parameter with least RMSE.

B. *Line plot with training data from 2013 to 2020, the predictions together with the confidence interval.*

```
# your code to perform the Arima train on data 2003 to 2020
train = create_df.iloc[:-1]
model_best = ARIMA(train, order=(1,1,1))
result_best = model_best.fit()

# Your code to predict for 2021 and 2022
preds_best=result_best.predict(start=2021,end=2022,dynamic=True)
```

```
# Plot the points and the prediction with its confidence interval
create_df.loc[2021] =[preds_best.iloc[0]]
create_df.loc[2022] =[preds_best.iloc[1]]
ci = 1.96 * np.std(create_df)/np.mean(create_df)

plt.plot(train,c="orange")
plt.plot(preds_best,c="Red")
plt.fill_between(create_df.index, np.squeeze(create_df-ci), np.squeeze(create_df+ci), color='b', alpha=.5)
plt.legend(["Training data","Forecasting","Confidence Interval"])
```

```
<matplotlib.legend.Legend at 0x7f70e3031210>
```



Part III - Self Reflection - Essay

1.Python's standard library offers a wide range of facilities and is very extensive which helps us to code efficiently. It provides standardised solutions to a lot of programming problems and it allows us to access system functionality such as file I/O which would be inaccessible to programmers otherwise. Standard library contains list for math related functions and functions that depend on operating system and can help to fetch data from web. And for special functions we need to use third party packages. Panda which stands for Python Data Analysis Library is an open-source python package which provides data analysis tools for labelled data which are easy to use and provides high performance. It helps us in data manipulation and is a perfect tool for data wrangling. It helps us operate with data frame objects containing rows and columns. Panda works as a foundation library for programmers. Panda library is not suitable for 3D matrix, and we need to move to NumPy to resolve that. We use NumPy package to facilitate math operations on arrays and to use them in vector forms to improve the performance reduce execution time. NumPy stores generic many dimensional data and provides tools to work with multi-dimensional array objects. It is helpful when working with Linear Algebra or Date Time. NumPy requires contiguous allocation of memory which is a drawback for this library. Matplotlib is a very essential plotting library which is helpful in embedding charts and plots into applications using object-oriented API. It provides us with a wide range of visualisations like histograms, scatter plots which can be created with just a little effort. To generate visualisations, we need to do a lot of coding, as it is a low-level library. (desai, 2019)

2. Big data can be stored using Hadoop distributed file system (HDFS), it is stored in clusters made from smaller blocks and these blocks are stored on internal disk drives. The data is duplicated twice so that if one system corrupts, the data can be retrieved from another device. It can store a large amount of data and provides better access to this information. The data can be retrieved and analysed using MapReduce. It is a framework that can process large amount of data. It works through two steps: Map and Reduce. The splitting and mapping of data is done by Map tasks and shuffling and reducing is done by reduce task. (vaidya, 2020) Hadoop is fast, flexible and resilient to failure. MapReduce is secure software to access the data and it requires authentication to use it. Hadoop is not fit to store small data and security is always a concern when dealing with such huge amount of data. The processing of data using MapReduce requires a lot of data to be shuffled over the networks.

3. Panda vs Spark

- Using Panda we can easily read csv files using read_csv() but Spark does not natively support csv we need to use separate spark-csv library.
- sparkDF.count() returns number of rows whereas pandasDF.count() gives no. of non-null observations for each column.
- In pandas we do not have to worry about data types as they are inferred but in spark the csv loaded files have default string type
- To view the data in tabular form we need to use pandasDF.head(), in spark we need to use sparkDF.show()
- Pandas library compute corrected standard deviation whereas spark computes uncorrected standard deviation, it is due to use of N-1 in denominator instead of N.
- In pandas new columns can be created using operator[], but in spark we need to use .withCoumn() to create columns as dataframe is immutable using spark. (Bourguignat, 2016)

4. Big Data term is used for datasets that are very large and complex, difficult to process using normal software. Big data has following characters: volume, velocity, variety, veracity and value. The huge volume of data collected also have a lot of personal/sensitive information of users which needs to be protected as unauthorised access to it can damage the reputation of individuals and firms. The data owners have to very careful with the storage and analysis of data, as the risk of breach is always imminent. (Anon., n.d.) Uber is a ridesharing company, a large number of drivers and customers use uber app on daily basis. The application stores all the personal, financial details of driver and customers, this huge amount of sensitive data is very difficult to manage and hackers continuously keeps attacking to breach the data and take advantage of the collected data. In 2016, two unauthorised persons accessed the uber data via 3rd party cloud service that uber uses to manage its data. The stolen data included names and driver's license no of more than 6 lakhs drivers working in USA. And the other details that were compromised included mobile numbers, email and names of riders and drivers. This breach if considered as an eligible data breach can allow the hackers to access customers information and harass them for money and put them in psychological distress. (employee, n.d.)

5. The big data can be secured using a lot of methods to protect sensitive information of customers. If the entity hires a third party to store its data, ask the provider to use efficient protection mechanisms to secure the data and it should be reviewed periodically to improve the security. Put a system in place to restrict the access to unauthorised persons so that only approved user has hand on information. The data stored on cloud should be encrypted so that sensitive data is protected, and it becomes hard for the hackers to retrieve the encrypted data. The data transfer among

company's server should be secure and not allow any outsiders control the transfer and it should be confidential. The data transfer and any breach attempts should be monitored regularly, and threat intelligence should be established to limit the damage. Uber uses two factor authorisations to secure its ridesharing application, users must sign-in using their id, passwords and it also randomly asks for the user's selfie to make sure that unauthorised person is not using the application. If the clicked image does not match the user's face, Uber prohibits the user from making further use of the application. So, its very hard for any unauthorised person to access the customer's information.

6. my thought about data science is that it is very interesting, but we have to focus a lot and need to understand the concept instead to facing problem in programming. Programming is just simple, but we have to give some time. For example, one of my team members is not from technical background but we did group study by sitting in library and that team member contributes very well in assignment. But if students are not from technical background, then do not worry about programming just understand the concept and discuss with friend and lectures and tutor about problem.