SIT742 Modern Data Science

Assignment 2

Team Members:

Student Name	Yash Ahuja	Priya Singh	Somya Kapoor
Student ID	219608443	219592426	219339951

Date: 26/05/2021

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

*A. The number of requests in weblog_df is **2530620***

1.2. Feature Selection

A. Data Description of ml_df.

	count	unique	top	freq
cs_method	2530620	6	GET	2525568
c_ip	2530620	69837	59.188.33.66	27935
cs_uri_stem	2530620	3616	/Tulip/common/common_style.aspx	121304
cs(User_Agent)	2530620	4653	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	594218
sc_status	2530620	12	200	1821993

Count denotes the number of total requests(records) for that attribute.

Unique denotes the number of unique requests for that attribute.

Top denotes the example value of the attribute which has highest frequency.

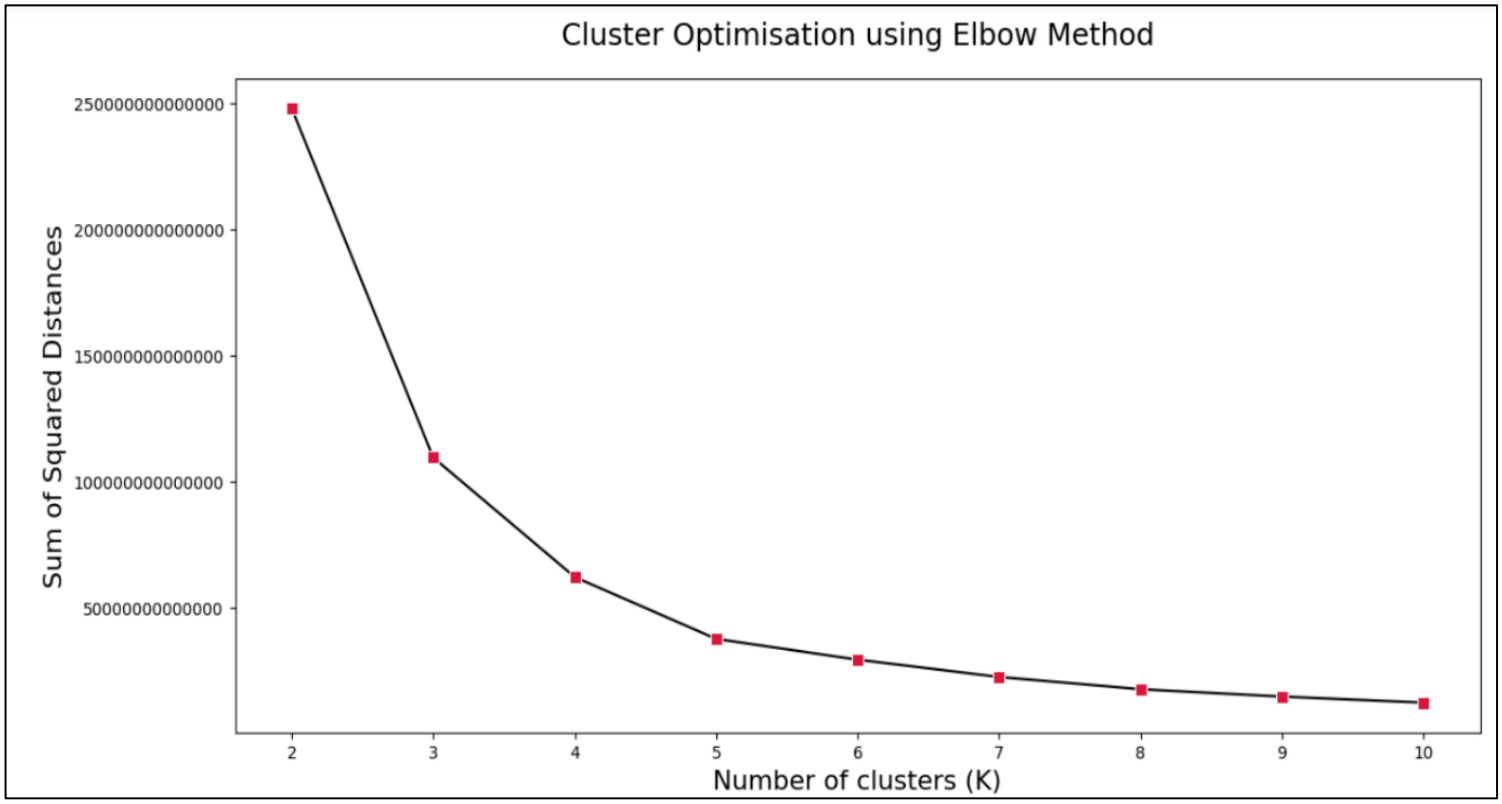
Frequency denotes the number of times the **top** value of the attribute has appeared in total number of requests.

B. *Top 5 rows of ml_df.*

	cs_method	c_ip	cs_uri_stem	cs(User_Agent)	sc_status
0	GET	218.102.231.100	/Tulip/common/en-us/images/topmenu_zh-hk.gif	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	200
1	GET	218.190.226.30	/Tulip/common/en-us/images/top_logo.gif	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	200
2	GET	203.241.147.26	/Tulip/common/en-us/images/sectionbanner_about...	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	304
3	GET	203.85.132.97	/Tulip/public/2899_2.jpg	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	200
4	GET	218.186.9.5	/Tulip/common/en-us/images/topmenu_zh-hk.gif	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	200

2. Unsupervised learning

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



From the above figure, it is evident that Optimal value of K for this dataset is **5** as it is the smallest value of K where WCSSD is low and increasing the number of clusters will not significantly reduce the WCSSD. A sharp line angle or an 'elbow' on the arm is identified at this point **K=5** implying the best value of K. Hence, the name of the method is elbow method.

WCSSD is Within Cluster Sum of Squared Distances, a key metric for cluster evaluation. A low WCSSD value indicates high similarity within clusters and therefore implying good clustering.

3. Supervised learning

3.1. Data Preparation

```
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)])

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

```
#Only 10% of the data is used in this part.
sl_df = sl_df.sample(fraction=0.1, seed=1)
print('Total Dataset Count: '+str(sl_df.count()))
```

```
Total Dataset Count: 253029
```

```
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|    18825|2741|       1385|            0|[18825.0,2741.0,1...|
|        0|    68103|2425|       1459|            0|[68103.0,2425.0,1...|
|        0|    41459|2087|       4238|            0|[41459.0,2087.0,4...|
+---------+---------+----+-----------+-------------+--------------------+
only showing top 3 rows
```

```
df_sl = df_temp.drop('cs_method', 'c_ip', 'cs_uri_stem', 'cs(User_Agent)')
df_sl.show(3)
```

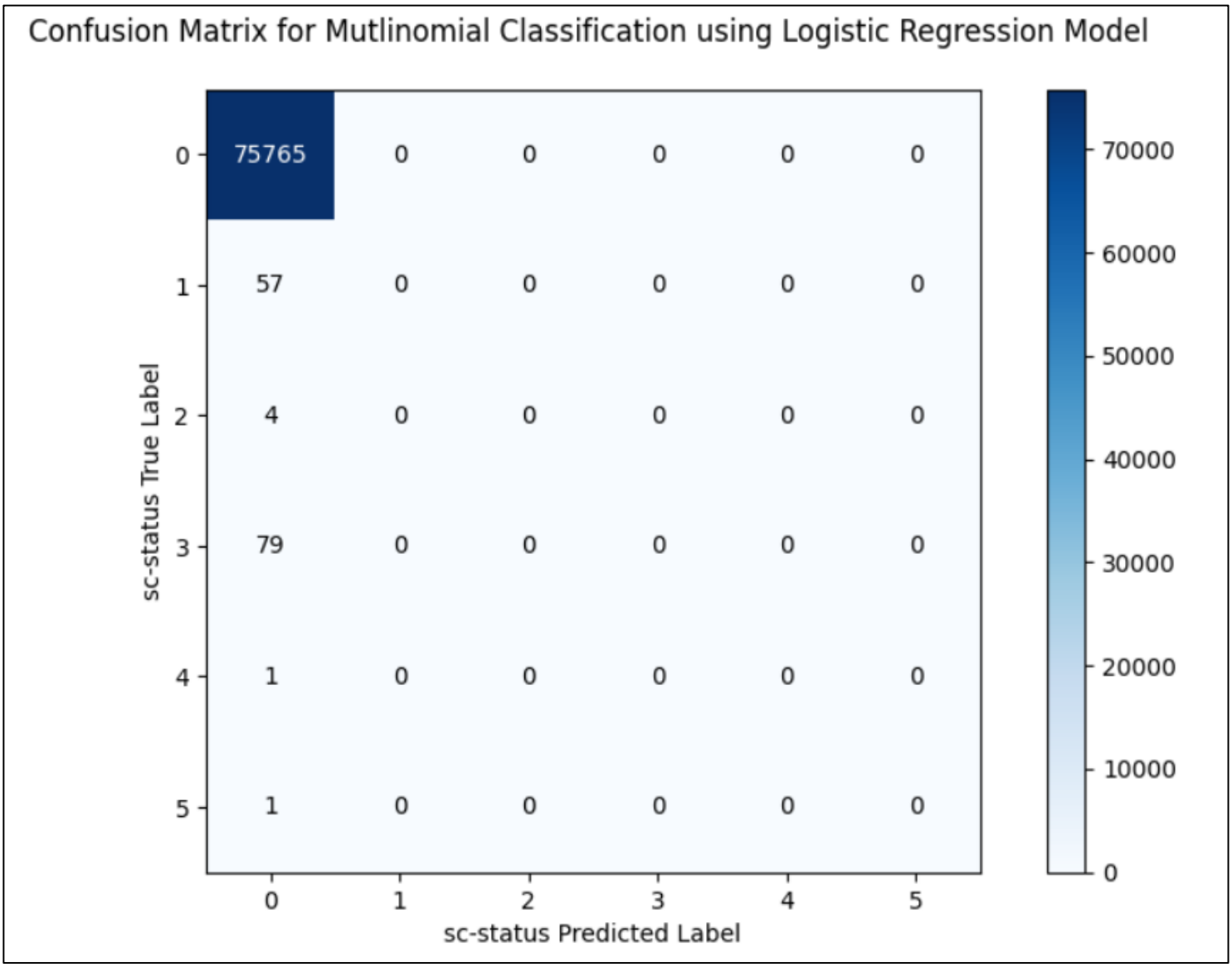
```
+---------+--------------------+
|sc_status|            features|
+---------+--------------------+
|        0|[18825.0,2741.0,1...|
|        0|[68103.0,2425.0,1...|
|        0|[41459.0,2087.0,4...|
+---------+--------------------+
only showing top 3 rows
```

3.2. Logistic Regression

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

Printing top 10 rows containing observations and predictions of sc_status leveraging multinomial classification on ml_df using logistic regression model:

```
+---------+----------+
|sc_status|prediction|
+---------+----------+
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
+---------+----------+
only showing top 10 rows
```



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

	precision	recall	f1-score	support
0	1.00	1.00	1.00	75765
1	0.00	0.00	0.00	57
2	0.00	0.00	0.00	4
3	0.00	0.00	0.00	79
4	0.00	0.00	0.00	1
5	0.00	0.00	0.00	1
accuracy			1.00	75907
macro avg	0.17	0.17	0.17	75907
weighted avg	1.00	1.00	1.00	75907

It shows the logistic regression model classification result using classification performance metrices such as precision, recall, f1-score.

Recall: True Positive / (True Positive + False Negative).

Out of all the items that are truly positive, how many were correctly classified as positive. Or simply, how many positive items were 'recalled' from the dataset.

Precision: True Positive / (True Positive + False Positive).

Out of all the items labeled as positive, how many truly belong to the positive class.

F1 Score: 2*((Precision*Recall)/(Precision + Recall)).

It is a measure to illustrate the balance between the Recall and the Precision. Therefore, also known as F Measure or F Score.

3.3. K-fold Cross Validation

A. Our code design and running results

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

# K = 2
# Parameters
# maxIter --> the number of iterations taken for the solvers to converge
maxIter_plist = [5, 10, 15]
# regParam --> regularization parameter
regParam_plist = [0.2, 0.4, 0.6]
# elasticNetParam --> elastic net regularization parameter
elasticNetParam_plist = [0, 0.1, 0.2]

# Developing parameter grid for k=2 fold cross validation by hyper tuning above parameters
parameter_grid = (ParamGridBuilder().addGrid(logr.maxIter, maxIter_plist)\
                  .addGrid(logr.regParam, regParam_plist)\
                  .addGrid(logr.elasticNetParam, elasticNetParam_plist).build())

# Creating K=2 fold cross validator
cross_valid = CrossValidator(estimator=logr, estimatorParamMaps=parameter_grid, evaluator=evaluator, numFolds=2)
# Cross validation model
cross_valid_model = cross_valid.fit(trainingData)
predictions = cross_valid_model.transform(testData)

# Best model Parameters
best_model = cross_valid_model.bestModel
best_maxIter = best_model._java_obj.getMaxIter()
best_regParam = best_model._java_obj.getRegParam()
best_elasticNetParam = best_model._java_obj.getElasticNetParam()
```

```
print('Best Model: "Maximum Iteration" parameter is= '+str(best_maxIter))
print('Best Model: "Regularization" parameter is= '+str(best_regParam))
print('Best Model: "Elastic Regularization" parameter is= '+str(best_elasticNetParam))

# Evaluating the best model
evaluator = MulticlassClassificationEvaluator(labelCol='sc_status', predictionCol='prediction')
print('Area Under the Curve for the best model is: '+str(evaluator.evaluate(predictions)))
print('Precision for the best model is: '+str(evaluator.evaluate(predictions, {evaluator.metricName:'weightedPrecision'})))
print('Recall for the best model is: '+str(evaluator.evaluate(predictions, {evaluator.metricName:'weightedRecall'})))
print('F1 score for the best model is: '+str(evaluator.evaluate(predictions, {evaluator.metricName:'f1'})))


# Printing results of the Best Model (top 10 rows containing observations and predictions of sc-status)
print('###############################################################################################')
print('Predictions of the Best Model i.e. Top 10 rows containing observations and predictions of sc-status:')
predictions.select('sc_status', 'prediction').show(10)
```

```
Best Model: "Maximum Iteration" parameter is= 5
Best Model: "Regularization" parameter is= 0.2
Best Model: "Elastic Regularization" parameter is= 0.0
Area Under the Curve for the best model is: 0.9971948103915026
Precision for the best model is: 0.9962620791343353
Recall for the best model is: 0.9981292897888205
F1 score for the best model is: 0.9971948103915026
##########################################################################################
Predictions of the Best Model i.e. Top 10 rows containing observations and predictions of sc-status:
+---------+----------+
|sc_status|prediction|
+---------+----------+
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
|        0|       0.0|
+---------+----------+
only showing top 10 rows
```

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

It is evident that the parameters used for training i.e. maxIter = 5, regParam= 0.2, elasticNetParam=0 are found to be the ones which have the best model performance after performing K=2 fold Cross Validation and that is why Area Under the Curve, Precision, Recall and F1 score metrics are same for training model and cross validation model.

4. Association Rule Mining

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

```
!pip install apyori
```

```
Collecting apyori
  Downloading https://files.pythonhosted.org/packages/5e/62/5ffde5c473ea4b033490617ec5caa80d59804875ad3c3c57c0976533a21a/apyori-1.1.2.tar.gz
Building wheels for collected packages: apyori
  Building wheel for apyori (setup.py) ... done
  Created wheel for apyori: filename=apyori-1.1.2-cp37-none-any.whl size=5975 sha256=212aa524bcd3c0e2872be6de63e326ef638bb47c78f6159a591f43060cc45d4e
  Stored in directory: /root/.cache/pip/wheels/5d/92/bb/474bbadbc8c0062b9eb168f69982a0443263f8ab1711a8cad0
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.2
```

```
# Taking sample (19,000 requests) from whole dataframe (df_ht) for association rule mining due to computational limitations
arm_df = df_ht.sample(n=19000, random_state=1)
# Total Attributes in dataframe
attributes = arm_df.columns
for col in attributes:
  print(col)
```

```
date
time
s_sitename
s_ip
cs_method
cs_uri_stem
s_port
c_ip
cs(User_Agent)
sc_status
sc_substatus
sc_win32_status
```

```
# Choosing appropriate attributes in order to make sense of the rules
arm_df = arm_df.loc[:,['cs_method','c_ip','cs_uri_stem','cs(User_Agent)','sc_status']]
chosen_attr = arm_df.columns
for col in chosen_attr:
  print(col)
```

```
cs_method
c_ip
cs_uri_stem
cs(User_Agent)
sc_status
```

Description of the above attributes:

- `cs_method:` The type of action which client tried to perform.
- `cd_uri_stem:` The resource accessed by the client.
- `c_ip:` Client IP address i.e. the IP address of the client who accessed the server.
- `cs(User-Agent):` The browser used by the client to access the web server.
- `sc_status:` Protocol Status i.e. the status of the server to client (sc) action (represented by a success or error code).

The above attributes are chosen based on their context as they may or may not have significant association in real-world.

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

# Resetting the index of arm_df
# arm_df = arm_df.reset_index(drop=True)

# Generating requests list
requests = []
for i in range(0, 19000):
  requests.append([str(arm_df.values[i,j]) for j in range(0, 5)])

print(requests)
```

```
'Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1)', '304'], ['GET', '61.202.225.129', '/Tulip/dining/ja/dining_terraceb
ar.aspx', 'Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.0;+.NET+CLR+1.1.4322)', '200'], ['GET', '219.78.58.179', '/Tul
ip/whatsnew/en-us/images/heading_fooddetail.gif', 'Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1)', '200'], ['GE
T', '203.169.153.162', '/promotion/footer2007.jpg', 'Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1)', '200'],
['GET', '222.148.118.143', '/Tulip/accommodation/images/photo_normal0.jpg', 'Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+N
T+5.1;+SV1)', '304'], ['HEAD', '202.108.23.56', '/Tulip/dining/zh-hk/images/sectionbanner_lobbylounge.swf', 'Mozilla/4.0+
(compatible;+MSIE+5.00;+Windows+98;+DigExt)', '200'], ['GET', '202.175.186.152', '/Tulip/aboutus/zh-hk/aboutus_motto.asp
x', 'Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+5.1)', '200'], ['GET', '218.252.240.94', '/Tulip/home/zh-hk/images/hom
e.swf', 'Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1)', '200'], ['GET', '219.78.113.203', '/Tulip/common/zh-c
n/images/sectionbanner_dining_off.gif', 'Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1)', '200'], ['GET', '81.35.186.
190', '/Tulip/dining/images/photo_tohlee1.jpg', 'Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1;+.NET+CLR+1.0.370
5;+.NET+CLR+1.1.4322)', '200'], ['GET', '219.73.51.7', '/Tulip/aboutus/en-us/aboutus_motto.aspx', 'Mozilla/4.0+(compatibl
e;+MSIE+6.0;+Windows+NT+5.1;+SV1;+.NET+CLR+1.1.4322)', '200'], ['GET', '219.78.153.187', '/Tulip/common/en-us/images/btn_
virtualtour.gif', 'Mozilla/5.0+(Windows;+U;+Windows+NT+5.1;+en-US;+rv:1.8.1.1)+Gecko/20061204+Firefox/2.0.0.1', '200'],
['GET', '61.93.39.249', '/Tulip/catering/zh-hk/images/btn_fountainsroom.gif', 'Mozilla/4.0+(compatible;+MSIE+6.0;+Windows
+NT+5.1;+SV1;+MEGAUPLOAD+1.0)', '200'], ['GET', '58.152.193.42', '/Tulip/dining/images/rightmenu_bg_sagano.gif', 'Mozill
a/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1;+.NET+CLR+1.1.4322)', '200'], ['GET', '203.218.96.87', '/Tulip/dining/zh
-hk/images/btn_whatsNew.gif', 'Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1;+.NET+CLR+1.1.4322;+InfoPath.1;+.NE
T+CLR+2.0.50727)', '200'], ['GET', '210.184.71.81', '/promotion/FP00807_1.jpg', 'Mozilla/4.0+(compatible;+MSIE+6.0;+Windo
ws+NT+5.1;+SV1;+.NET+CLR+1.1.4322)', '304'], ['GET', '210.139.29.89', '/Tulip/common/ja/images/sectionbanner_recruitment_
```

```
# Applying the apriori algorithm
rules = apriori(requests, min_support=0.001, min_confidence=0.2, min_lift=3, max_length=2)

# Printing out the rules
output = list(rules)
#print(output)
for i in range(len(output)):
  print('################################################################################')
  print(i)
  print(output[i])
```

```
################################################################################
0
RelationRecord(items=frozenset({'206', '/Tulip/dining/images/tohLee.pdf'}), support=0.0011052631578947368, ordered_statistic
s=[OrderedStatistic(items_base=frozenset({'/Tulip/dining/images/tohLee.pdf'}), items_add=frozenset({'206'}), confidence=1.0,
lift=115.15151515151516)])
################################################################################
1
RelationRecord(items=frozenset({'206', '/Tulip/public/7036_4.pdf'}), support=0.0012105263157894737, ordered_statistics=[Orde
redStatistic(items_base=frozenset({'/Tulip/public/7036_4.pdf'}), items_add=frozenset({'206'}), confidence=0.766666666666666
7, lift=88.28282828282829)])
################################################################################
2
RelationRecord(items=frozenset({'/aspnet_client/system_web/1_1_4322/WebUIValidation.js', '404'}), support=0.0020526315789473
684, ordered_statistics=[OrderedStatistic(items_base=frozenset({'/aspnet_client/system_web/1_1_4322/WebUIValidation.js'}), i
tems_add=frozenset({'404'}), confidence=1.0, lift=60.12658227848102)])
################################################################################
3
RelationRecord(items=frozenset({'/favicon.ico', '404'}), support=0.008526315789473684, ordered_statistics=[OrderedStatistic
(items_base=frozenset({'/favicon.ico'}), items_add=frozenset({'404'}), confidence=1.0, lift=60.12658227848102), OrderedStati
stic(items_base=frozenset({'404'}), items_add=frozenset({'/favicon.ico'}), confidence=0.5126582278481012, lift=60.1265822784
8101)])
################################################################################
4
RelationRecord(items=frozenset({'/robots.txt', '404'}), support=0.0012105263157894737, ordered_statistics=[OrderedStatistic
(items_base=frozenset({'/robots.txt'}), items_add=frozenset({'404'}), confidence=1.0, lift=60.12658227848102)])
################################################################################
```

```
5
RelationRecord(items=frozenset({'Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1;+.NET+CLR+1.1.4322;+InfoPath.1)', '2
02.181.238.254'}), support=0.001631578947368421, ordered_statistics=[OrderedStatistic(items_base=frozenset({'202.181.238.25
4'}), items_add=frozenset({'Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1;+.NET+CLR+1.1.4322;+InfoPath.1)'}), confi
dence=0.40789473684210525, lift=26.006711409395972)])
################################################################################
6
RelationRecord(items=frozenset({'210.184.71.81', 'Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+98)'}), support=0.0022631578947
36842, ordered_statistics=[OrderedStatistic(items_base=frozenset({'210.184.71.81'}), items_add=frozenset({'Mozilla/4.0+(comp
atible;+MSIE+6.0;+Windows+98)'}), confidence=0.24431818181818177, lift=17.650362945039745)])
################################################################################
7
RelationRecord(items=frozenset({'210.184.71.81', 'Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1)'}), support=0.002263157
894736842, ordered_statistics=[OrderedStatistic(items_base=frozenset({'210.184.71.81'}), items_add=frozenset({'Mozilla/4.0+
(compatible;+MSIE+6.0;+Windows+NT+5.1)'}), confidence=0.24431818181818177, lift=3.508726723012437)])
################################################################################
8
RelationRecord(items=frozenset({'218.255.20.100', 'Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1)'}), support=0.001
0526315789473684, ordered_statistics=[OrderedStatistic(items_base=frozenset({'218.255.20.100'}), items_add=frozenset({'Mozil
la/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1)'}), confidence=1.0, lift=5.157437567861021)])
################################################################################
9
RelationRecord(items=frozenset({'304', 'Mozilla/4.0+(compatible;)'}), support=0.0015263157894736842, ordered_statistics=[Ord
eredStatistic(items_base=frozenset({'Mozilla/4.0+(compatible;)'}), items_add=frozenset({'304'}), confidence=0.76315789473684
21, lift=3.0033140016570012)])
################################################################################
10
RelationRecord(items=frozenset({'404', 'Mozilla/4.0+(compatible;+MSIE+6.0;+Win32)'}), support=0.0012105263157894737, ordered
_statistics=[OrderedStatistic(items_base=frozenset({'Mozilla/4.0+(compatible;+MSIE+6.0;+Win32)'}), items_add=frozenset({'40
4'}), confidence=0.32857142857142857, lift=19.755877034358047)])
```

```
# Printing out the rules in another way to make it easily interpretable
for rule in output:
  frozen_set = rule[0]
  elements = [x for x in frozen_set]
  print('################################################################################')
  print('Rule: ' + elements[0] + ' ' + '-->' + ' ' + elements[1])
  print('Support: ' + str(rule[1]))
  print('Confidence: ' + str(rule[2][0][2]))
  print('Lift: ' + str(rule[2][0][3]))
```

```
################################################################################
Rule: 206 --> /Tulip/dining/images/tohLee.pdf
Support: 0.0011052631578947368
Confidence: 1.0
Lift: 115.15151515151516
################################################################################
Rule: 206 --> /Tulip/public/7036_4.pdf
Support: 0.0012105263157894737
Confidence: 0.7666666666666667
Lift: 88.28282828282829
################################################################################
Rule: /aspnet_client/system_web/1_1_4322/WebUIValidation.js --> 404
Support: 0.0020526315789473684
Confidence: 1.0
Lift: 60.12658227848102
################################################################################
Rule: /favicon.ico --> 404
Support: 0.008526315789473684
Confidence: 1.0
Lift: 60.12658227848102
################################################################################
Rule: /robots.txt --> 404
Support: 0.0012105263157894737
Confidence: 1.0
Lift: 60.12658227848102
################################################################################
```

```
################################################################################
Rule: Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1;+.NET+CLR+1.1.4322;+InfoPath.1) --> 202.181.238.254
Support: 0.001631578947368421
Confidence: 0.40789473684210525
Lift: 26.006711409395972
################################################################################
Rule: 210.184.71.81 --> Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+98)
Support: 0.002263157894736842
Confidence: 0.24431818181818177
Lift: 17.650362945039745
################################################################################
Rule: 210.184.71.81 --> Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1)
Support: 0.002263157894736842
Confidence: 0.24431818181818177
Lift: 3.508726723012437
################################################################################
Rule: 218.255.20.100 --> Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1)
Support: 0.0010526315789473684
Confidence: 1.0
Lift: 5.157437567861021
################################################################################
Rule: 304 --> Mozilla/4.0+(compatible;)
Support: 0.0015263157894736842
Confidence: 0.7631578947368421
Lift: 3.0033140016570012
################################################################################
Rule: 404 --> Mozilla/4.0+(compatible;+MSIE+6.0;+Win32)
Support: 0.0012105263157894737
Confidence: 0.32857142857142857
Lift: 19.755877034358047
```

```
# Converting rules to pandas dataframe
def rulestodf(output):
    left_hand_side = [tuple(item[2][0][0])[0] for item in output]
    right_hand_side = [tuple(item[2][0][1])[0] for item in output]
    support = [item[1] for item in output]
    confidence = [item[2][0][2] for item in output]
    lift = [item[2][0][3] for item in output]
    return list(zip(left_hand_side, right_hand_side, support, confidence, lift))

m_result_df = pd.DataFrame(rulestodf(output), columns=['Left Hand Side', 'Right Hand Side', 'Support', 'Confidence', 'Lift'])
m_result_df
```

]:

	Left Hand Side	Right Hand Side	Support	Confidence	Lift
0	/Tulip/dining/images/tohLee.pdf	206	0.001105	1.000000	115.151515
1	/Tulip/public/7036_4.pdf	206	0.001211	0.766667	88.282828
2	/aspnet_client/system_web/1_1_4322/WebUIValida...	404	0.002053	1.000000	60.126582
3	/favicon.ico	404	0.008526	1.000000	60.126582
4	/robots.txt	404	0.001211	1.000000	60.126582
5	202.181.238.254	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	0.001632	0.407895	26.006711
6	210.184.71.81	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+98)	0.002263	0.244318	17.650363
7	210.184.71.81	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	0.002263	0.244318	3.508727
8	218.255.20.100	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	0.001053	1.000000	5.157438
9	Mozilla/4.0+(compatible;)	304	0.001526	0.763158	3.003314
10	Mozilla/4.0+(compatible;+MSIE+6.0;+Win32)	404	0.001211	0.328571	19.755877

B. Our findings on association rule mining results

```
# Reporting the top five association rules based on lift metric
arm_result_df.nlargest(5, columns='Lift')
```

	Left Hand Side	Right Hand Side	Support	Confidence	Lift
0	/Tulip/dining/images/tohLee.pdf	206	0.001105	1.000000	115.151515
1	/Tulip/public/7036_4.pdf	206	0.001211	0.766667	88.282828
2	/aspnet_client/system_web/1_1_4322/WebUIValida...	404	0.002053	1.000000	60.126582
3	/favicon.ico	404	0.008526	1.000000	60.126582
4	/robots.txt	404	0.001211	1.000000	60.126582

It is evident from the above output of top five association rules that resource used by the client i.e. cs-uri-stem attribute has significant association with the protocol status (the status of the server to client action, usually represented by a code) i.e. sc-status attribute which makes sense in the real-world.

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 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

Screenshot of our code for generating citation2003-2021.csv.

```
# Importing web crawling libraries
import requests
from bs4 import BeautifulSoup

# Fixing the size of the output cell
from IPython.display import Javascript
display(Javascript('''google.colab.output.setIframeHeight(0, true, {maxHeight: 500})'''))

url = 'https://scholar.google.com/citations?user=dqwjm-0AAAAJ&hl=en#d=gsc_md_hist'
response = requests.get(url)
soup = BeautifulSoup(response.content, 'lxml')
print(soup.prettify())
```

```
<IPython.core.display.Javascript object>

<!DOCTYPE html>
<html>
 <head>
  <title>
   Gang Li - Google Scholar
  </title>
  <meta content="text/html;charset=utf-8" http-equiv="Content-Type"/>
  <meta content="IE=Edge" http-equiv="X-UA-Compatible"/>
  <meta content="always" name="referrer"/>
  <meta content="width=device-width,initial-scale=1,minimum-scale=1,maximum-scale=2" name="viewport"/>
  <meta content="telephone=no" name="format-detection"/>
  <link href="/favicon.ico" rel="shortcut icon"/>
  <link href="http://scholar.google.com/citations?user=dqwjm-0AAAAJ&amp;hl=en" rel="canonical"/>
  <meta content="Deakin University - Cited by 4.227 - data privacy - data mining - tourism - hospitality" name="descripti
on"/>
  <meta content="Deakin University - Cited by 4.227 - data privacy - data mining - tourism - hospitality" property="og:de
scription"/>
  <meta content="Gang Li" property="og:title"/>
```

From the above output, it is evident that citation information from 2003-2021 is stored in **div class='gsc_md_hist_b'**

```
# Fixing the size of the output cell
from IPython.display import Javascript
display(Javascript('''google.colab.output.setIframeHeight(0, true, {maxHeight: 500})'''))

# Getting the citation information from div class='gsc_md_hist_b'
barchart = soup.find('div', class_='gsc_md_hist_b')
print(barchart.prettify())
```

```
<div class="gsc_md_hist_b">
 <span class="gsc_g_t" style="right:579px">
  2003
 </span>
 <span class="gsc_g_t" style="right:547px">
  2004
 </span>
 <span class="gsc_g_t" style="right:515px">
  2005
 </span>
 <span class="gsc_g_t" style="right:483px">
  2006
 </span>
 <span class="gsc_g_t" style="right:451px">
  2007
 </span>
 <span class="gsc_g_t" style="right:419px">
  2008
 </span>
```

```
# Extracting Year and Citation information from 2003-2021
year = []
Citation = []

for row in barchart.find_all('span', class_='gsc_g_t'):
  year.append(row.contents[0])

for row in barchart.find_all('span', class_='gsc_g_al'):
  Citation.append(row.contents[0])

# Creating dataframe containing years and citations from 2003-2021

df_list = {'year':year, 'Citation':Citation}
create_df = pd.DataFrame(df_list, columns=['year','Citation'])
print('Total number of years from 2003-2021: '+str(len(create_df)))
create_df
```

```
Total number of years from 2003-2021: 19
```

	year	Citation
0	2003	15
1	2004	34
2	2005	17
3	2006	11
4	2007	33
5	2008	41
6	2009	57
7	2010	68
8	2011	105
9	2012	131
10	2013	170
11	2014	251
12	2015	290
13	2016	340
14	2017	385
15	2018	452
16	2019	583
17	2020	842
18	2021	322

```
# Saving the citation dataframe in csv format to virtual machine
create_df.to_csv('Citation2003-2021.csv', index=False)
# Downloading the csv file to local machine
from google.colab import files
files.download('Citation2003-2021.csv')
```

6. Train Arima to predict the 2018 to 2020 citation

6.1. Train Arima Model

Screenshot of our code for Arima Training.

```
# Loading the citation data from Citation2003-2021.csv generated from create_df
citations = pd.read_csv('Citation2003-2021.csv', parse_dates=True, index_col=0, header=0, squeeze=True)
X = citations.values

# Changing the string datatype of citations to float
X = X.astype('float32')

# Split the citations data into train (year 2003 to 2017) and test (year 2018 to 2020)
trainingData, testData = X[0:15], X[15:18]

print('Training Data count: '+str(len(trainingData)))
print('Test Data count: '+str(len(testData)))
```

```
Training Data count: 15
Test Data count: 3
```

6.2. Predicting the citation and Calculate the RMSE

Screenshot of our code for predicting and display RMSE (root mean square error).

```
# Installing statsmodels
!pip install "statsmodels==0.11.1"
```

```
Requirement already satisfied: statsmodels==0.11.1 in /usr/local/lib/python3.7/dist-packages (0.11.1)
Requirement already satisfied: pandas>=0.21 in /usr/local/lib/python3.7/dist-packages (from statsmodels==0.11.1) (1.1.5)
Requirement already satisfied: numpy>=1.14 in /usr/local/lib/python3.7/dist-packages (from statsmodels==0.11.1) (1.19.5)
Requirement already satisfied: scipy>=1.0 in /usr/local/lib/python3.7/dist-packages (from statsmodels==0.11.1) (1.4.1)
Requirement already satisfied: patsy>=0.5 in /usr/local/lib/python3.7/dist-packages (from statsmodels==0.11.1) (0.5.1)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.21->statsmodels==0.11.
1) (2018.9)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.21->statsmod
els==0.11.1) (2.8.1)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from patsy>=0.5->statsmodels==0.11.1) (1.15.0)
```

```python
# Predicting the citation and saving it to variable preds. Also, print the output of the confidence interval(95%) here as wel

import warnings
warnings.filterwarnings('ignore')

# Importing libraries for ARIMA
from statsmodels.tsa.arima.model import ARIMA

history = [y for y in trainingData]
predictions = list()
confidence_interval = []
year = 2018

for t in range(len(testData)):
  arima_model = ARIMA(history, order=(1,1,1)) # parameter order = (p=1,d=1,q=1)
  arima_model_fit = arima_model.fit()
  results = arima_model_fit.get_forecast()
  preds = results.predicted_mean[0]
  predictions.append(preds)
  observ = testData[t]
  history.append(observ)
  ci = results.conf_int(0.05)
  confidence_interval.append(ci[0])
  print('#########################################################')
  print('Predicted Citations for year %d = %d, Expected Citations for year %d = %d' %(year, int(preds), year, int(observ)))
  print('95%% Confidence Interval for year %d: %.2f to %.2f' %(year, ci[0,0], ci[0,1]))
  year += 1
```

```
#########################################################
Predicted Citations for year 2018 = 427, Expected Citations for year 2018 = 452
95% Confidence Interval for year 2018: 388.02 to 466.58
#########################################################
Predicted Citations for year 2019 = 505, Expected Citations for year 2019 = 583
95% Confidence Interval for year 2019: 465.75 to 544.61
#########################################################
Predicted Citations for year 2020 = 682, Expected Citations for year 2020 = 842
95% Confidence Interval for year 2020: 629.86 to 736.01
```

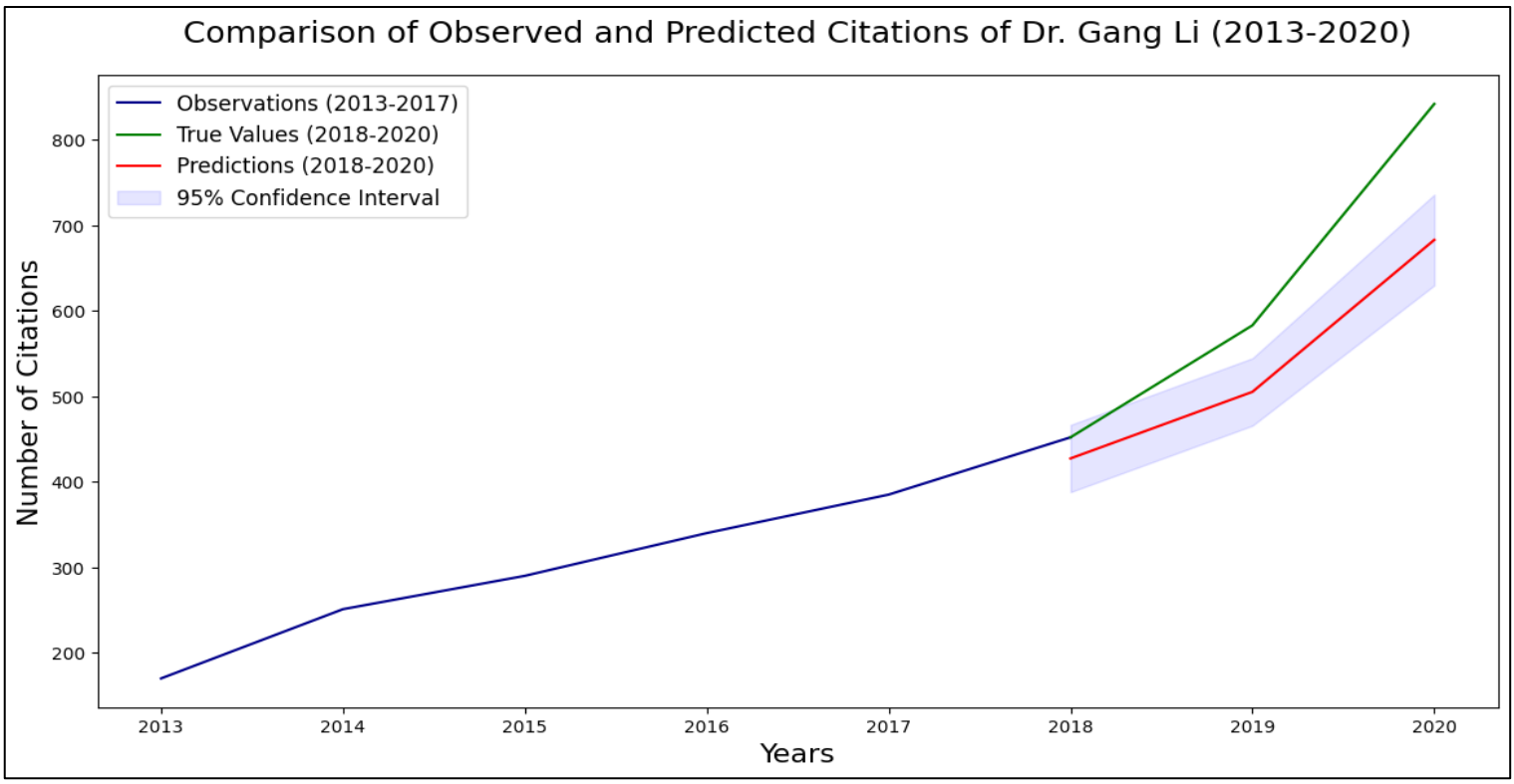
```python
# Print the error below by comparing the test and preds:
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import numpy as np
from math import sqrt

rmse = sqrt(mean_squared_error(testData, predictions))
print('Root Mean Square Error (RMSE) for the test = %.2f' %(rmse))
```

```
Root Mean Square Error (RMSE) for the test = 103.23
```

6.3. Draw the visualization to compare

Screenshot of our line plot, along with its code.

```
# Plotting the citations from 2013-2017, predicted citations (2018-2020) against actual citations (2018-2020) and also the co
train_data = range(2013,2019)
train_list = list(trainingData[10:]) + [testData[0]]
test_data = range(2018,2021)
plt.figure(figsize=[14,7], dpi=100)
plt.plot(train_data, train_list, color='darkblue')
plt.plot(test_data, testData, color='green' )
plt.plot(test_data, predictions, color='red')
plt.fill_between(test_data, np.array(confidence_interval)[:,0], np.array(confidence_interval)[:,1], alpha=0.1, color='b')
plt.legend(labels=['Observations (2013-2017)', 'True Values (2018-2020)', 'Predictions (2018-2020)', '95% Confidence Interval
plt.title('Comparison of Observed and Predicted Citations of Dr. Gang Li (2013-2020)', pad=20, fontsize=18)
plt.xlabel('Years', fontsize=16)
plt.ylabel('Number of Citations', fontsize=16)
plt.show()
```



7. Parameter selection and Year 2021 and 2022 Prediction

7.1. Grid Search

Screenshot of our code for grid search.

```
# Conducting Grid Search on Parameters

#trainingData -> 2003-2017
#testData -> 2018-2020

history = [y for y in trainingData]
predictions = list()
confidence_interval = []
RMSE = []
Parameter = []
p = list(range(1,3)) #[1,2]
q = list(range(1,3)) #[1,2]
d = list(range(1,3)) #[1,2]

# Walk-Forward Validation Approach

for j1 in p:
  for j2 in d:
    for j3 in q:
      for t in range(len(testData)):
        arima_model = ARIMA(history, order=(j1,j2,j3)) # parameter order = (p=j1,d=j2,q=j3)
        arima_model_fit = arima_model.fit()
        results = arima_model_fit.forecast()
        preds = results[0]
        predictions.append(preds)
        observ = testData[t]
        history.append(observ)
      rmse = sqrt(mean_squared_error(testData, predictions))
      history = [y for y in trainingData]
      predictions = list()
      RMSE.append(rmse)
      param_list = [j1,j2,j3]
      Parameter.append(param_list)
      print('Root Mean Square Error (RMSE) for the test with parameter p=%d, d=%d, q=%d is = %.2f' %(j1, j2, j3, rmse))
```

```
min_rmse = min(RMSE)
ind = RMSE.index(min_rmse)
optimal_param = Parameter[ind]
print('#########################################################')
print(f"The optimal parameter set with minimum RMSE value = {round(min_rmse,2)} is p = {optimal_param[0]}, d = {optimal_param
```

```
Root Mean Square Error (RMSE) for the test with parameter p=1, d=1, q=1 is = 103.23
Root Mean Square Error (RMSE) for the test with parameter p=1, d=1, q=2 is = 103.70
Root Mean Square Error (RMSE) for the test with parameter p=1, d=2, q=1 is = 97.64
Root Mean Square Error (RMSE) for the test with parameter p=1, d=2, q=2 is = 94.38
Root Mean Square Error (RMSE) for the test with parameter p=2, d=1, q=1 is = 92.95
Root Mean Square Error (RMSE) for the test with parameter p=2, d=1, q=2 is = 102.64
Root Mean Square Error (RMSE) for the test with parameter p=2, d=2, q=1 is = 106.22
Root Mean Square Error (RMSE) for the test with parameter p=2, d=2, q=2 is = 104.28
#########################################################
The optimal parameter set with minimum RMSE value = 92.95 is p = 2, d = 1, q=1
```

```
# Generate the seach-results.csv and print the top 6 rows
Results = pd.DataFrame({'RMSE':RMSE,'Parameter':Parameter})
Results.head(6)
```

	RMSE	Parameter
0	103.228218	[1, 1, 1]
1	103.702932	[1, 1, 2]
2	97.640804	[1, 2, 1]
3	94.379291	[1, 2, 2]
4	92.949608	[2, 1, 1]
5	102.636291	[2, 1, 2]

```
# Saving the Results dataframe in csv format to virtual machine
Results.to_csv('Search-results.csv', index=False)
# Downloading the csv file to local machine
from google.colab import files
files.download('Search-results.csv')
```

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

A. Displaying the best parameter values

```
# Optimal Parameter Set
print(optimal_param)
```

```
[2, 1, 1]
```

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

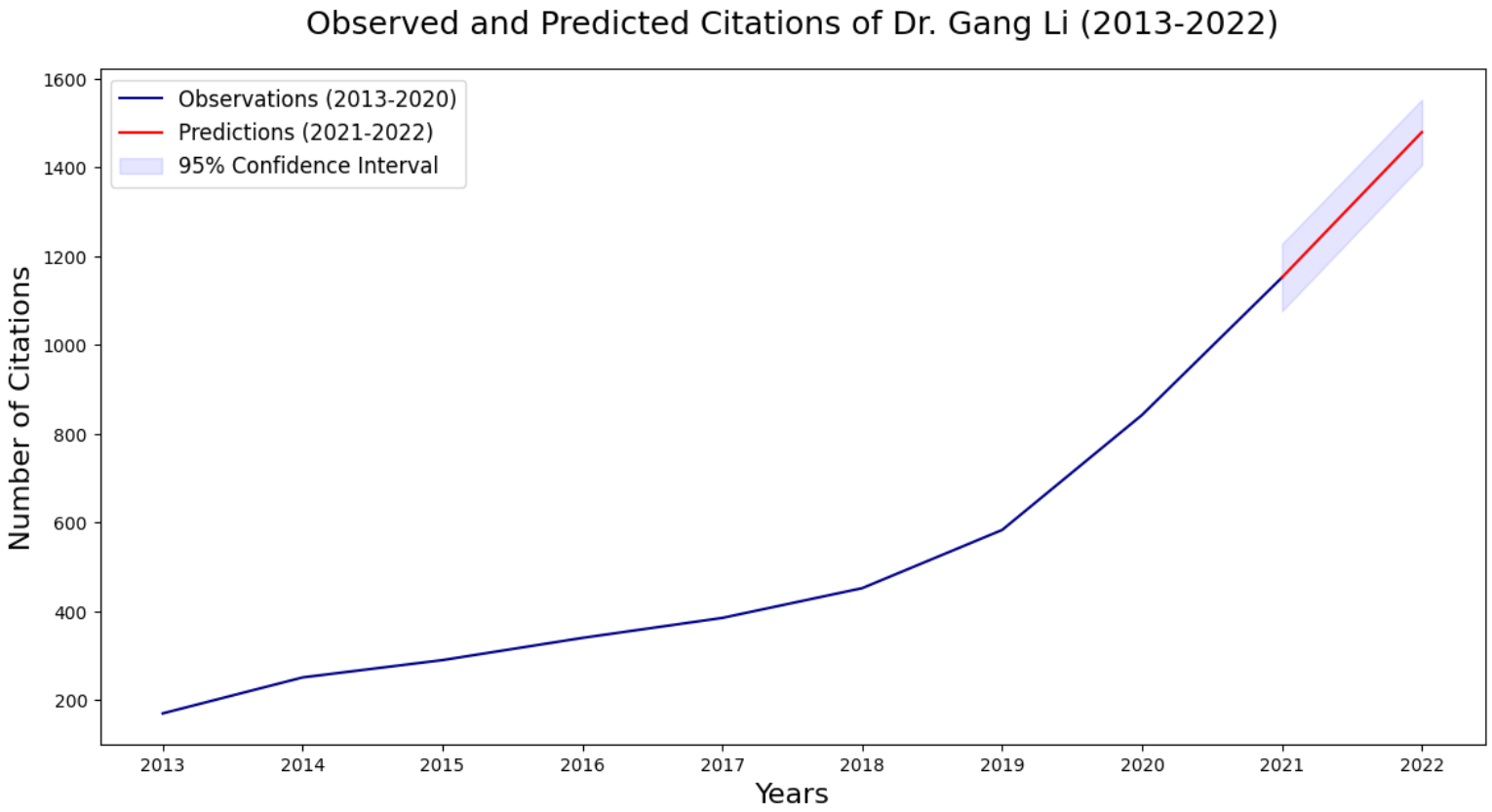
```
# Training Data from year 2003 to 2020
train_new = citations.values[:-1]

# Performing the Arima train on data from 2003 to 2020
history = [y for y in train_new]
predictions = list()
confidence_interval = []
year = 2021

for t in (range(2)):
  arima_model = ARIMA(history, order=(optimal_param[0],optimal_param[1],optimal_param[2])) # optimal parameter order -> (p=2,
  arima_model_fit = arima_model.fit()
  output = arima_model_fit.get_forecast()
  preds = output.predicted_mean[0]
  predictions.append(preds)
  history.append(preds)
  ci = output.conf_int(0.05)
  confidence_interval.append(ci[0])
  print('#####################################################')
  print('Predicted Citations for year %d = %d' %(year, int(preds)))
  print('95%% Confidence Interval for year %d: %.2f to %.2f' %(year, ci[0,0], ci[0,1]))
  year += 1
```

```
#####################################################
Predicted Citations for year 2021 = 1151
95% Confidence Interval for year 2021: 1075.84 to 1227.16
#####################################################
Predicted Citations for year 2022 = 1478
95% Confidence Interval for year 2022: 1405.26 to 1552.10
```

```
# Plotting the observed citations from 2013-2020, predicted citations (2020-2021) and also the corresponding confidence inter
data = range(2013,2023)
obs_data = range(2013,2022)
obs_list = list(train_new[10:]) + [predictions[0]]
pred_data = range(2021,2023)
plt.figure(figsize=[14,7], dpi=100)
plt.plot(obs_data, obs_list, color='darkblue')
plt.plot(pred_data, predictions, color='red' )
plt.fill_between(pred_data, np.array(confidence_interval)[:,0], np.array(confidence_interval)[:,1], alpha=0.1, color='b')
plt.xticks(data)
plt.legend(labels=['Observations (2013-2020)', 'Predictions (2021-2022)', '95% Confidence Interval'], fontsize=12)
plt.title('Observed and Predicted Citations of Dr. Gang Li (2013-2022)', pad=20, fontsize=18)
plt.xlabel('Years', fontsize=16)
plt.ylabel('Number of Citations', fontsize=16)
plt.show()
```

95% Confidence Interval for year 2021*: **1075.84 to 1227.16***

95% Confidence Interval for year 2022*: **1405.26 to 1552.10***



Part III - Self Reflection - Essay

1. What are the Python packages that you find useful in manipulating and analysing Big data? You can briefly analyse their advantages and disadvantages:

***Pandas*:** Pandas is open-source python package. Pandas is used for data analysis and data structures and operations for manipulating numerical tables and time series.

a) **Advantages**:
   i) Extensive features set.
   ii) Makes data flexible and customizable.

b) **Disadvantages:**
   i) Steep Learning curve.
   ii) Complex syntax.

Numpy: It is used for scientific and numerical computation in Python. It allows working on large datasets and storage.

a) **Advantages:**
   i) Takes less memory space and provide better runtime speed.
   ii) Support Vectorized operations
b) **Disadvantages:**
   i) Requires contagious allocation of memory.

Scipy: Scipy is scientific computation library in Python. It provides utility functions for optimization, stats, and signal processing.

a) **Advantages:**
   i) SciPy contains various versions of the linear algebra modules and other numerical algorithms.
b) **Disadvantages:**
   i) Scipy is lower in computation than Numpy.

Matplotlib: It is python library used for 2d graph such as bar graph,histogram, pie chart for plotting.

a) **Advantages:**
   i) Easy to use for basic plots such as histogram, pie chart etc.

b) **Disadvantages:**
   ii) Reliant on other packages to work such as Numpy.

2. What are the Big data platforms that can help storing, retrieving, and analysing the big data? What are their advantages and disadvantages?

Some of the Big data platforms that can store, retrieve, and analyse the data are described as follows:

Apache Hadoop: Apache Hadoop is java-based software used to store large amount of data in cluster and run them parallelly across several nodes. The storage system of Hadoop is known as HDFS – Hadoop Database File System. It divides the data into small sets and process them in various nodes.

Advantages: Scalable, flexible, fault-tolerant, robust

Disadvantages: Not ideal for smaller datasets, security issues

NoSql: NoSQL is used to deal with unstructured datasets. These datasets are stored with no schema; however, it gives better performance in storing huge amount of data and analysing them.

Advantages: Low cost (open source), easily scalable, can store unstructured data (such as text documents)

Disadvantages: Lack of standardization, insignificant community support

Hive: Hive is also one database management system in BigData. It provides query options like SOL query and primarily used for data mining purposes.

Advantages: Data may be queries by several users at the same time, easy implementation of ETL jobs

Disadvantages: Used extensively for online analytical processing systems rather than online transaction processing systems.

Datawrapper: Data wrapper is open-source software for data visualization to generate charts and analysis on the datasets. It is very device friendly, fast, and interactive tool to depict the visualization of large datasets.

3. Compare and contrast the Python data analytical packages and their Spark packages.

Two spark packages are:

1) Apache Spark
2) PySpark

Apache Spark: Apache Spark is computing engine; it contains set of libraries which helps in parallel computing on computer clusters. It is open source. It is used for large scale data processing

Pyspark: Pyspark is python API, it is collaboration of apache spark and python, It is also open source and used for cluster computing framework. It is created and licenced under Apache Spark Foundation. It used library called Py4j, an API written in Python. It is especially used in big data.

Python data analytical packages: Python has standard library that supports a wide variety of functionalities like databases, automation, text processing, scientific computing. Python packages are generally easy to use, high performance and used in AI. Some of the python packages are Pandas, Numpy, SciPy, Seaborn, TensorFlow etc.

4. What are your opinions on the privacy issues in the Big data era? Any example to further illustrate the risks?

There are many benefits from Big Data analytics, but also have a massive potential for exposure that could result in anything. Mentioned below are some of the ways to prevent the risks in big data.

Some ways to prevent the exposure risk in big data are:

a) Govern Your Data Access

We live in an age where a mail truck full of bank account statements getting hi-jacked is unlikely, but a data warehouse full of social security numbers or intellectual property data getting hacked into is very prevalent. To combat these faceless culprits, you should take these steps to improve your understanding of who has access to your treasured data:

- Assess current access requirements and processes, key systems, or applications containing large volumes of data within your company environment. This is not just limited to internal employees, but also any third parties who may be receiving or using the data.
- Define baseline access requirements for these key systems or applications containing large datasets. This includes who should have access and what business justification should warrant access.

- Establish access controls for these big data stores such as strong authentication, and approvals from data owners before granting access. Or, reduce the number of users by appointing a central "librarian" to control access to data stores.
- Perform ongoing monitoring of user access against the baseline requirements. This will help you to proactively identify deviations from normal access and quickly address insufficient controls.

b) Limit the Data Use, Collection and Storage

Organizations often collect and hoard information that is not really required or even used, perhaps thinking there might be a need for it someday. As defined in the AICPA Generally Accepted Privacy Principles (GAPP), basic privacy principles such as "collection limited to identified purpose" and "use, retention, and disposal" urge organizations to:

- Review current data residing in big data stores and determine the business need for collecting, using, and storing them. If you are not using the data for legitimate business reasons, you should stop collecting it, as it only introduces more risk exposure.
- Understand the upstream and downstream data flow of your big data store. Sometimes the information is "fed" to us by a third party. If you want to stop receiving unnecessary data from the source, you need to first understand where it comes from.
- Follow the retention policy or schedule and diligently archive data into an approved archival solution. This will not only help you to reduce data related breaches and comply with retention requirements but will also improve capacity management of your networks and systems.

c) Leverage Technologies

Additionally, relying on technologies to help us secure big data is imperative. Protecting your data through end-to-end encryption or tokenization will help to minimize data from being understood by unauthorized people. However, consider these basic impacts of technology implementation on general accessibility and availability of data:

- Performance impact on data access. A few seconds may not seem to be a deal breaker, but if there are multiple databases and multiple applications across multiple platforms involved, then the sum of that lead time might be unacceptable to the end users or critical business processes.
- Business process impact. Utilizing tokenization to substitute a sensitive data element that cannot be mathematically reversed is a popular way to reduce Payment Card Information (PCI) Data Security Standard (DSS) scope. With the use of a token instead of payment card data, it might alter some of the business processes handling of payment card data.

- Complexity associated with key management. Without effective key management policies and processes, the encryption is no more secure than handing your keys over to the inmates.

Big data does not always come with supersized risks. You can reduce big data breaches by defining access requirements; limiting the collection, use, or storage of data to only support your business need; and applying technical controls to protect data from intruders.

5. What are the methods you think could help to solve the privacy issues on big data? Please list any successful implemented method.

Below are some privacy issues and how to solve them on Big data:

Data privacy best practices for big data

There are certain strategies organizations can use to protect big data. Several of the best practices for maintaining the privacy of big data include:

a) Employ real-time monitoring.

Since a privacy issue can happen at any moment, organizations should find a solution that monitors data in real-time. This way, they will be aware of a problem as soon as it happens and can take appropriate steps to resolve it right away.

b) Implement homomorphic encryption.

Homomorphic encryption is a form of encryption that allows users to compute data without decrypting it first. This form of encryption should be implemented to store and process information in the cloud to prevent organizations from revealing private information to outside vendors.

c) Avoid collecting too much data.

Only the data that is necessary should be collected. An organization may not need the Social Security numbers of their customers; customer login usernames and passwords may only be necessary. Organizations should consider deleting any personal information that is not needed to best protect customer data privacy.

d) Prevent internal threats.

Organizations are also exposed to internal privacy risks from angry or simply uninformed employees. Therefore, it is essential to educate all employees on best practices for ensuring data privacy like changing passwords frequently and logging off unused computers.

6. Any other thoughts about data science, or suggestions to future students (or teaching team) about this unit.

After completing the SIT742 Modern Data Science, through experiences we can suggest various things for future students. Some of them are:

Students who do not have python background, may find it difficult to understand advanced machine learning libraries and implantation of the concepts of data science. Thus, we recommend prerequisite of any python related unit or Machine learning unit as a mandatory unit for this unit.

Also, unit can arrange some extra online help hub sessions as refresher for the unit.

For teaching team, we suggest increasing the tutorial time and include the time for assessments related doubts. This will not only improve the understanding of assessments among the students but also improve the performance in the same. We highly recommend the teaching team to include every topic from the assessment to be covered in the tutorials, for example web crawling and other such topics used in assessment 2. Proper guidance can be provided through links and study material to study and understand those topics instead of doing research online.