SIT742 Modern Data Science

Assignment 2

Team Members:

Student Name	Mirna Arivalagan	Talia Sachi Sugiyama Martinez	Yanhong Wang
Student ID	220142881	218631894	219578053

Date: 22nd 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.

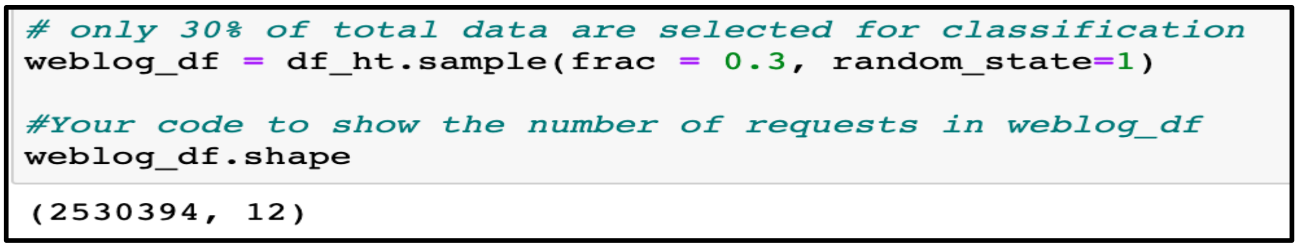
1. Data ETL

1.1. Load Data

Please add descriptions of the following contents by yourselves.

A. *The number of requests in weblog_df.*

There was a total of 2530394 requests in the weblog_df data frame



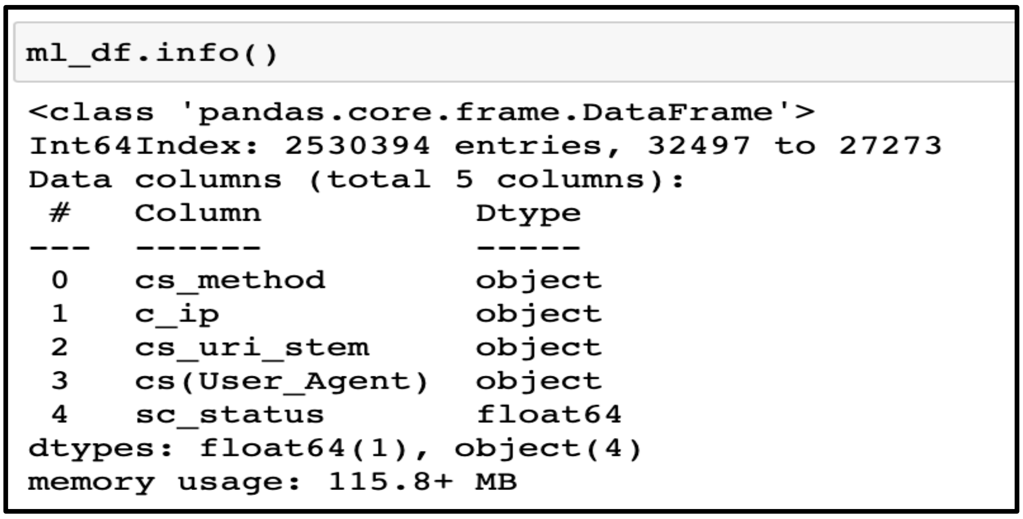
```
# only 30% of total data are selected for classification
weblog_df = df_ht.sample(frac = 0.3, random_state=1)

#Your code to show the number of requests in weblog_df
weblog_df.shape
```

```
(2530394, 12)
```

1.2. Feature Selection

Please add description of the following contents by yourselves.

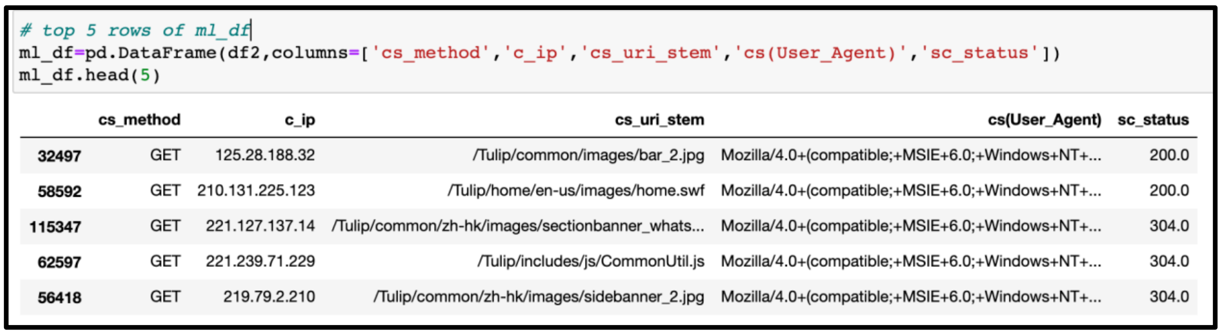
A. *Data Description of ml_df.*



```
ml_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2530394 entries, 32497 to 27273
Data columns (total 5 columns):
 #   Column           Dtype
---  ------           -----
 0   cs_method        object
 1   c_ip             object
 2   cs_uri_stem      object
 3   cs(User_Agent)   object
 4   sc_status        float64
dtypes: float64(1), object(4)
memory usage: 115.8+ MB
```

B. *Top 5 rows of ml_df.*



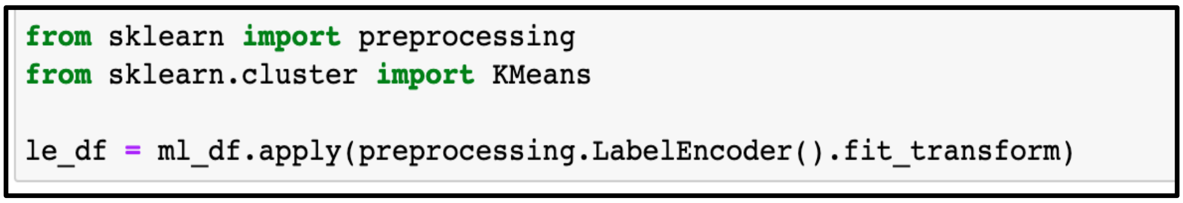
```
# top 5 rows of ml_df
ml_df=pd.DataFrame(df2,columns=['cs_method','c_ip','cs_uri_stem','cs(User_Agent)','sc_status'])
ml_df.head(5)
```

	cs_method	c_ip	cs_uri_stem	cs(User_Agent)	sc_status
32497	GET	125.28.188.32	/Tulip/common/images/bar_2.jpg	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	200.0
58592	GET	210.131.225.123	/Tulip/home/en-us/images/home.swf	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	200.0
115347	GET	221.127.137.14	/Tulip/common/zh-hk/images/sectionbanner_whats...	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	304.0
62597	GET	221.239.71.229	/Tulip/includes/js/CommonUtil.js	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	304.0
56418	GET	219.79.2.210	/Tulip/common/zh-hk/images/sidebanner_2.jpg	Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+...	304.0

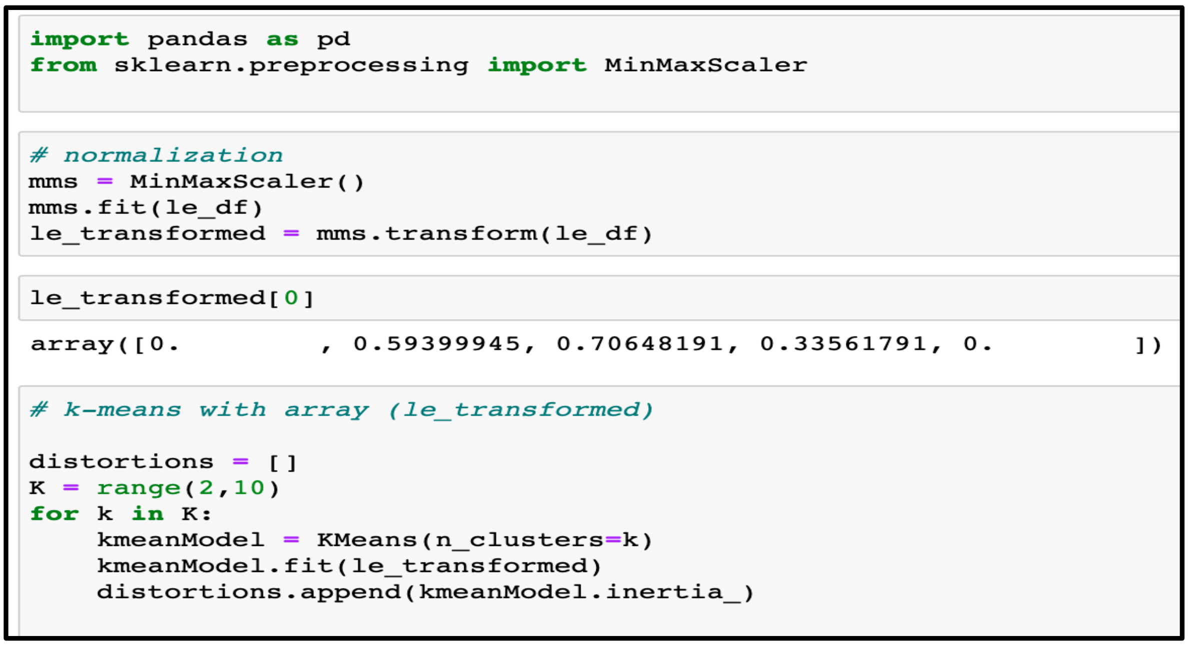
2. Unsupervised learning

Please add description of the following contents by yourselves.

A. *Display the clustering result using Accuracy.*



```
from sklearn import preprocessing
from sklearn.cluster import KMeans

le_df = ml_df.apply(preprocessing.LabelEncoder().fit_transform)
```



```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
```

```
# normalization
mms = MinMaxScaler()
mms.fit(le_df)
le_transformed = mms.transform(le_df)
```

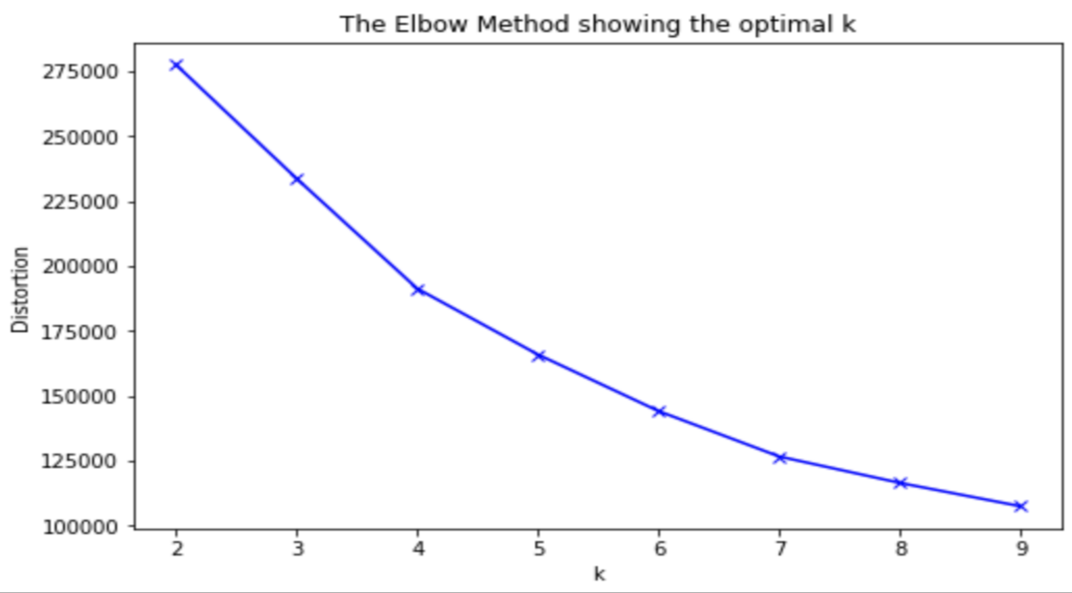
```
le_transformed[0]
```

```
array([0.        , 0.59399945, 0.70648191, 0.33561791, 0.        ])
```

```
# k-means with array (le_transformed)

distortions = []
K = range(2,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(le_transformed)
    distortions.append(kmeanModel.inertia_)
```

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

```
# visulization clusters
plt.figure(figsize=(8,5))
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```

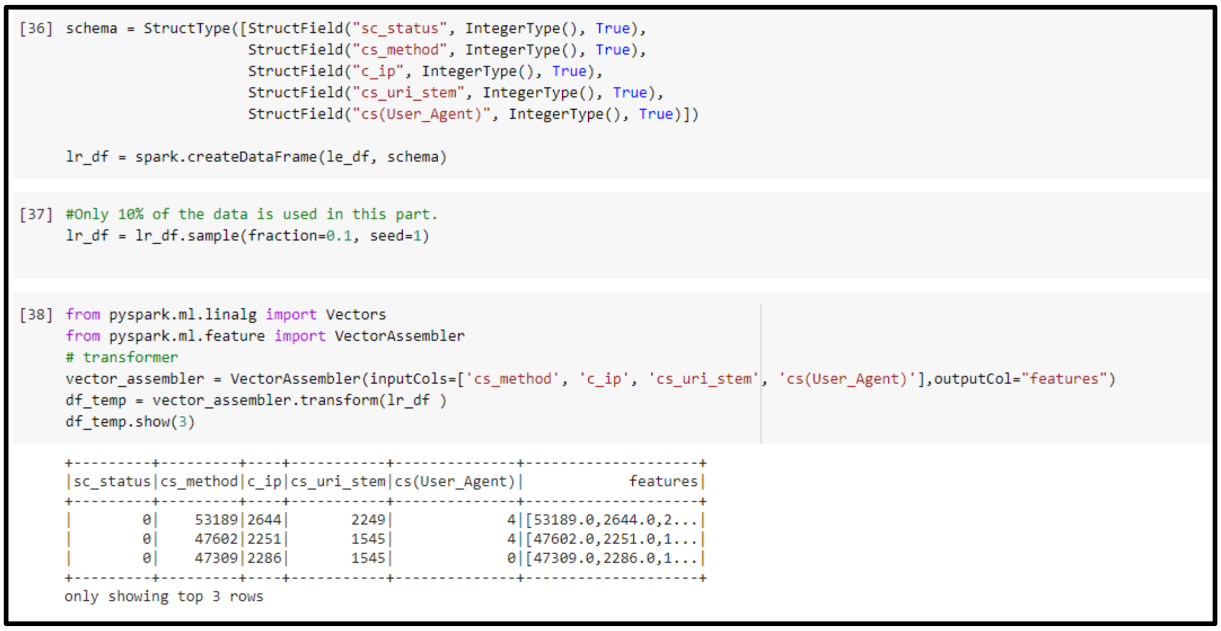


Clustering is the unsupervised method of grouping data that groups similar data together to detect any anomalies with the dataset. K-means clustering is the most common technique for cluster detection. As clustering methods cannot deal with nominal data, a labelEncode pre-processing step needs be done prior to the clustering task. LabelEncode transforms nominal variables into numeric variables. MinMaxScale is applied to normalize the data, this ensures that all the data is in the same range. In this unsupervised learning task, we have applied a K-means range between 2-10. The line chart that has been plotted identifies the sharp angle, which is known as the elbow, this elbow indentation indicates the best value of k, and in this case, the best value for K is 4, which means the data can be segmented into 4 clusters.

3. Supervised learning

3.1. Data Preparation

Please add descriptions by yourselves.

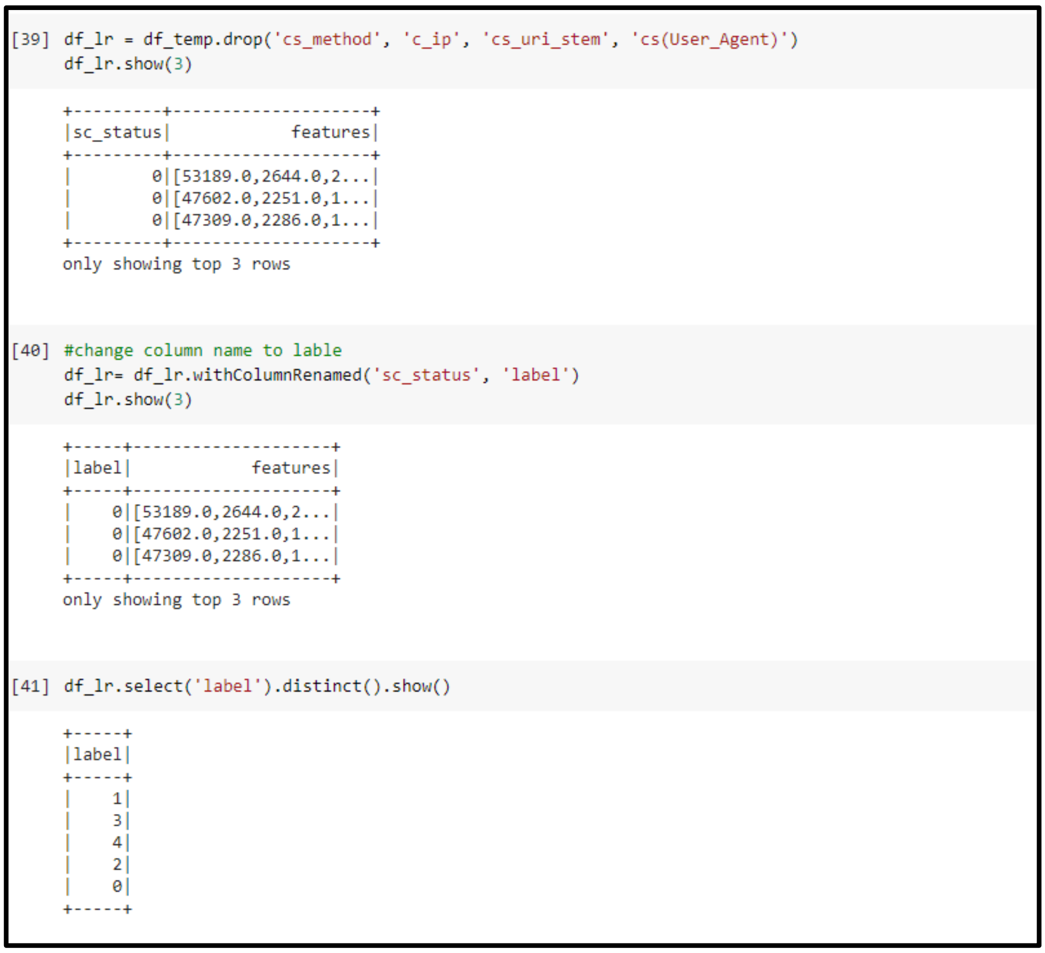


```
[36] schema = StructType([StructField("sc_status", IntegerType(), True),
                          StructField("cs_method", IntegerType(), True),
                          StructField("c_ip", IntegerType(), True),
                          StructField("cs_uri_stem", IntegerType(), True),
       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

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

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

```
[38] 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(lr_df )
     df_temp.show(3)
```

```
     +---------+---------+----+-----------+--------------+--------------------+
     |sc_status|cs_method|c_ip|cs_uri_stem|cs(User_Agent)|            features|
     +---------+---------+----+-----------+--------------+--------------------+
     |        0|    53189|2644|       2249|             4|[53189.0,2644.0,2...|
     |        0|    47602|2251|       1545|             4|[47602.0,2251.0,1...|
     |        0|    47309|2286|       1545|             0|[47309.0,2286.0,1...|
     +---------+---------+----+-----------+--------------+--------------------+
     only showing top 3 rows
```



```
[39] df_lr = df_temp.drop('cs_method', 'c_ip', 'cs_uri_stem', 'cs(User_Agent)')
     df_lr.show(3)
```

```
     +---------+--------------------+
     |sc_status|            features|
     +---------+--------------------+
     |        0|[53189.0,2644.0,2...|
     |        0|[47602.0,2251.0,1...|
     |        0|[47309.0,2286.0,1...|
     +---------+--------------------+
     only showing top 3 rows
```

```
[40] #change column name to lable
     df_lr= df_lr.withColumnRenamed('sc_status', 'label')
     df_lr.show(3)
```

```
     +-----+--------------------+
     |label|            features|
     +-----+--------------------+
     |    0|[53189.0,2644.0,2...|
     |    0|[47602.0,2251.0,1...|
     |    0|[47309.0,2286.0,1...|
     +-----+--------------------+
     only showing top 3 rows
```

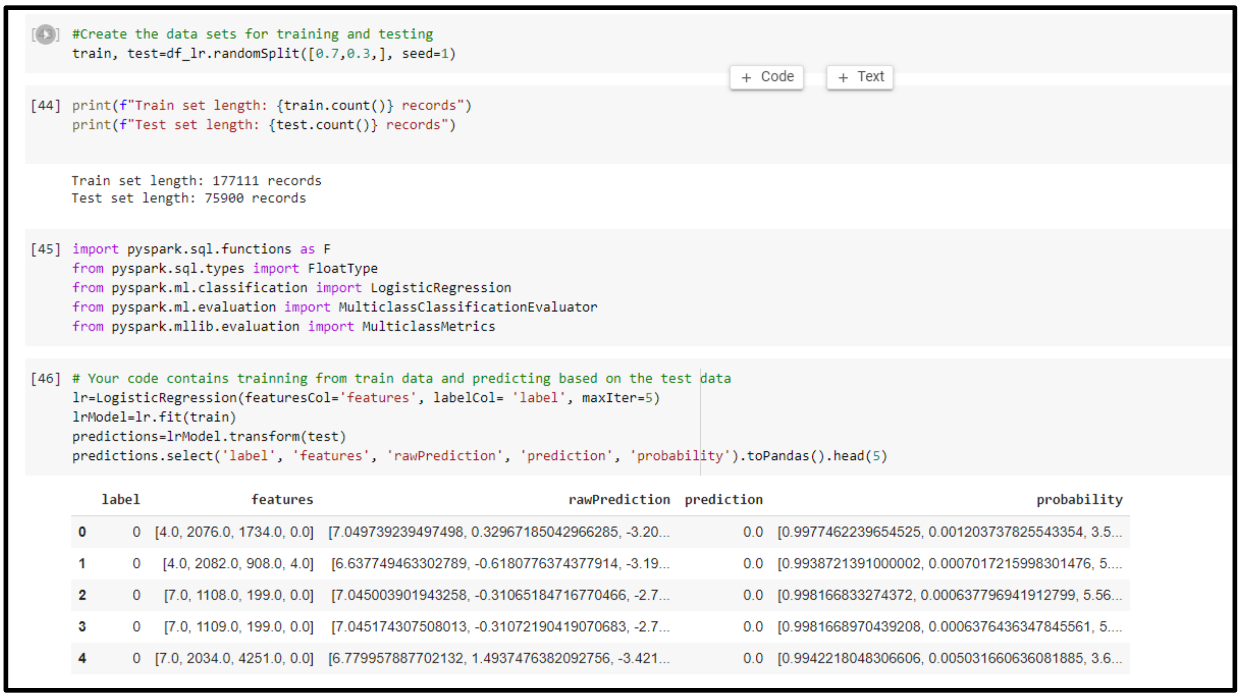
```
[41] df_lr.select('label').distinct().show()
```

```
     +-----+
     |label|
     +-----+
     |    1|
     |    3|
     |    4|
     |    2|
     |    0|
     +-----+
```

3.2. Logistic Regression

Please add description of the following contents by yourselves.

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



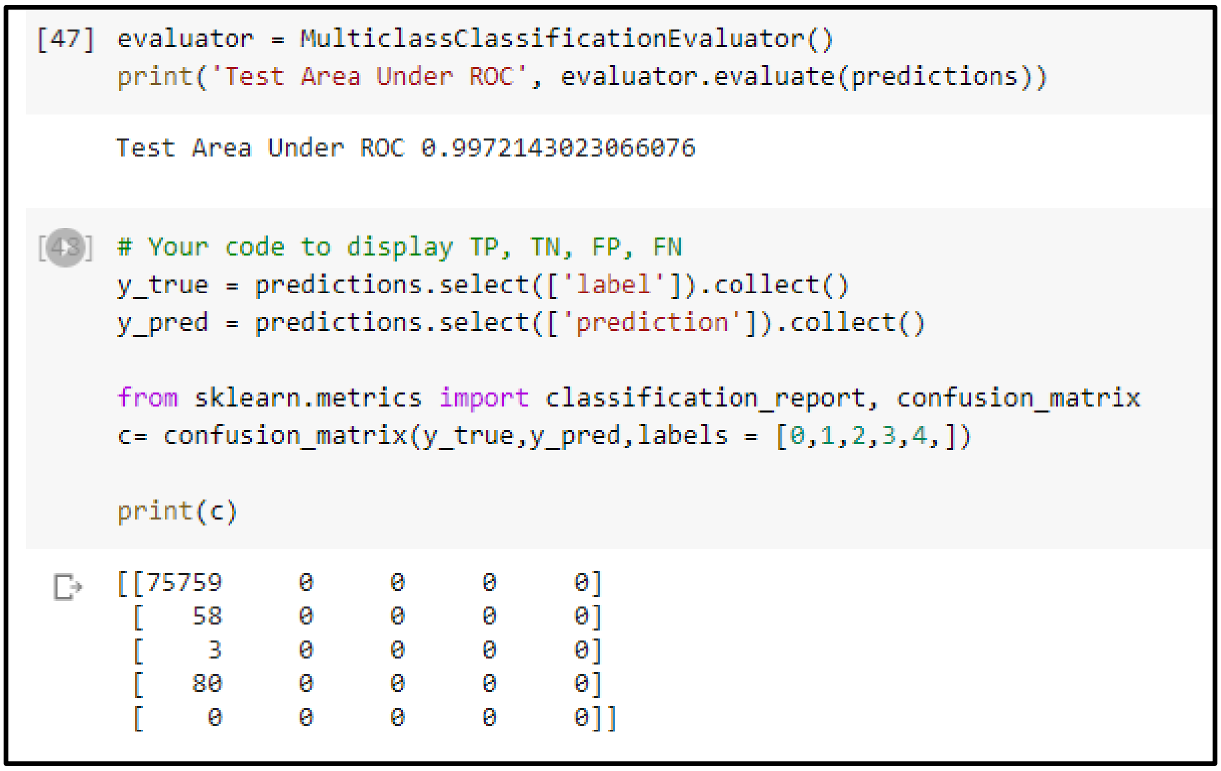
```
[ ] #Create the data sets for training and testing
    train, test=df_lr.randomSplit([0.7,0.3,], seed=1)
```

```
[44] print(f"Train set length: {train.count()} records")
     print(f"Test set length: {test.count()} records")

     Train set length: 177111 records
     Test set length: 75900 records
```

```
[45] import pyspark.sql.functions as F
     from pyspark.sql.types import FloatType
     from pyspark.ml.classification import LogisticRegression
     from pyspark.ml.evaluation import MulticlassClassificationEvaluator
     from pyspark.mllib.evaluation import MulticlassMetrics
```

```
[46] # Your code contains trainning from train data and predicting based on the test data
     lr=LogisticRegression(featuresCol='features', labelCol= 'label', maxIter=5)
     lrModel=lr.fit(train)
     predictions=lrModel.transform(test)
     predictions.select('label', 'features', 'rawPrediction', 'prediction', 'probability').toPandas().head(5)
```

	label	features	rawPrediction	prediction	probability
0	0	[4.0, 2076.0, 1734.0, 0.0]	[7.049739239497498, 0.32967185042966285, -3.20...	0.0	[0.9977462239654525, 0.001203737825543354, 3.5...
1	0	[4.0, 2082.0, 908.0, 4.0]	[6.637749463302789, -0.6180776374377914, -3.19...	0.0	[0.9938721391000002, 0.0007017215998301476, 5....
2	0	[7.0, 1108.0, 199.0, 0.0]	[7.045003901943258, -0.31065184716770466, -2.7...	0.0	[0.998166833274372, 0.000637796941912799, 5.56...
3	0	[7.0, 1109.0, 199.0, 0.0]	[7.045174307508013, -0.31072190419070683, -2.7...	0.0	[0.9981668970439208, 0.0006376436347845561, 5....
4	0	[7.0, 2034.0, 4251.0, 0.0]	[6.779957887702132, 1.4937476382092756, -3.421...	0.0	[0.9942218048306606, 0.005031660636081885, 3.6...

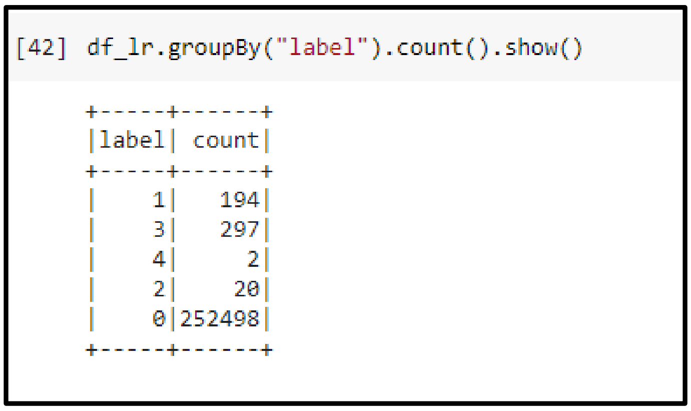


```
[47] evaluator = MulticlassClassificationEvaluator()
     print('Test Area Under ROC', evaluator.evaluate(predictions))

     Test Area Under ROC 0.9972143023066076
```

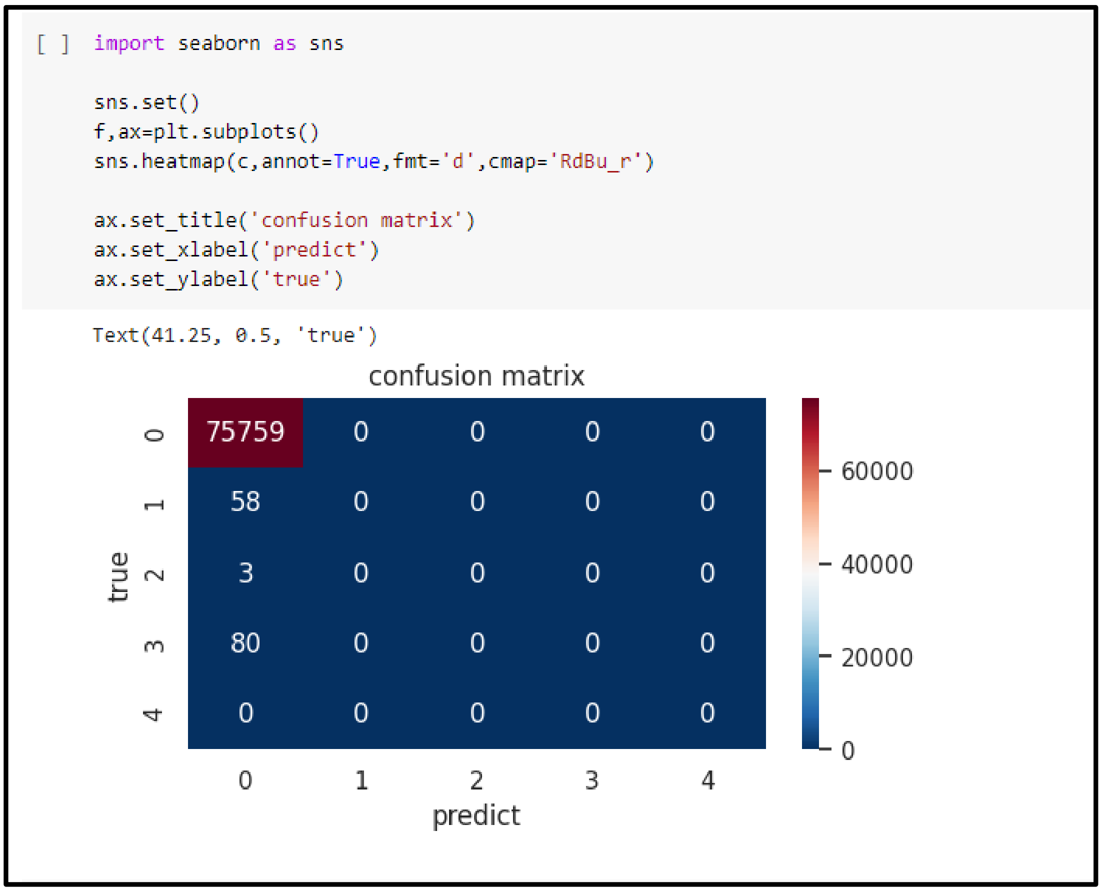
```
[48] # Your code to display TP, TN, FP, FN
     y_true = predictions.select(['label']).collect()
     y_pred = predictions.select(['prediction']).collect()

     from sklearn.metrics import classification_report, confusion_matrix
     c= confusion_matrix(y_true,y_pred,labels = [0,1,2,3,4,])

     print(c)

     [[75759     0     0     0     0]
      [   58     0     0     0     0]
      [    3     0     0     0     0]
      [   80     0     0     0     0]
      [    0     0     0     0     0]]
```

There were 757559 points in the first class (label 0). Out of these, our model was successful in identifying all of those correctly in label 0. From the total sample of 75900 99.8% were classified as first class (label 0), and they were true positives. However, we can see that 58 who were meant to be labelled as 1 were also classified as 0, 3 that were meant to be level 2 were also classified and 80 that were meant to be classified as 3 were classified as 0.

This shows us that our data being skewed with most of our data labelled 0, meant that the model predicted no other classification correctly.



```
[42] df_lr.groupBy("label").count().show()

     +-----+------+
     |label| count|
     +-----+------+
     |    1|   194|
     |    3|   297|
     |    4|     2|
     |    2|    20|
     |    0|252498|
     +-----+------+
```

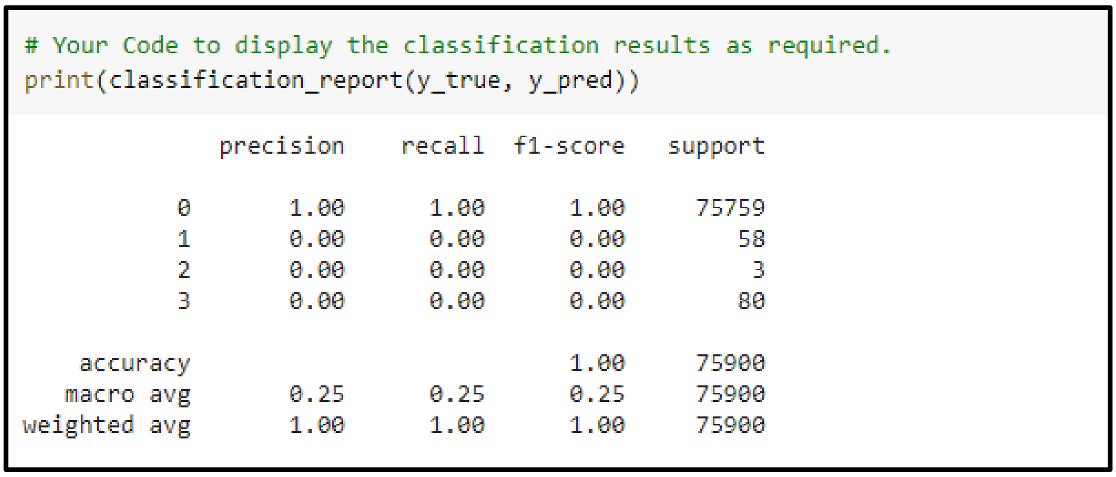


```
[ ] import seaborn as sns

    sns.set()
    f,ax=plt.subplots()
    sns.heatmap(c,annot=True,fmt='d',cmap='RdBu_r')

    ax.set_title('confusion matrix')
    ax.set_xlabel('predict')
    ax.set_ylabel('true')

    Text(41.25, 0.5, 'true')
```

true \ predict	0	1	2	3	4
0	75759	0	0	0	0
1	58	0	0	0	0
2	3	0	0	0	0
3	80	0	0	0	0
4	0	0	0	0	0

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

We can see a similar outcome in our classification report. Where for level 0 predictions we have 1.00 in all precision, recall and f score. Whilst for level 1,2,3 the result is 0.



```
# Your Code to display the classification results as required.
print(classification_report(y_true, y_pred))
```

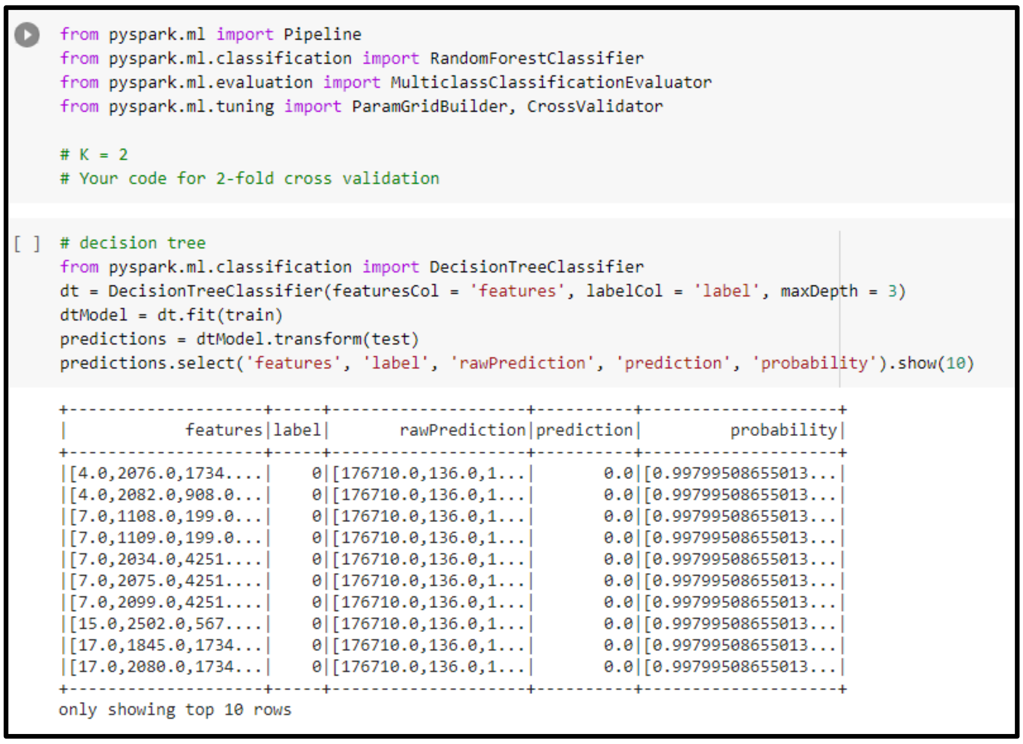
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     75759
           1       0.00      0.00      0.00        58
           2       0.00      0.00      0.00         3
           3       0.00      0.00      0.00        80

    accuracy                           1.00     75900
   macro avg       0.25      0.25      0.25     75900
weighted avg       1.00      1.00      1.00     75900
```

3.3. K-fold Cross Validation

Please add description of the following contents by yourselves.

A. Your code design and running results:

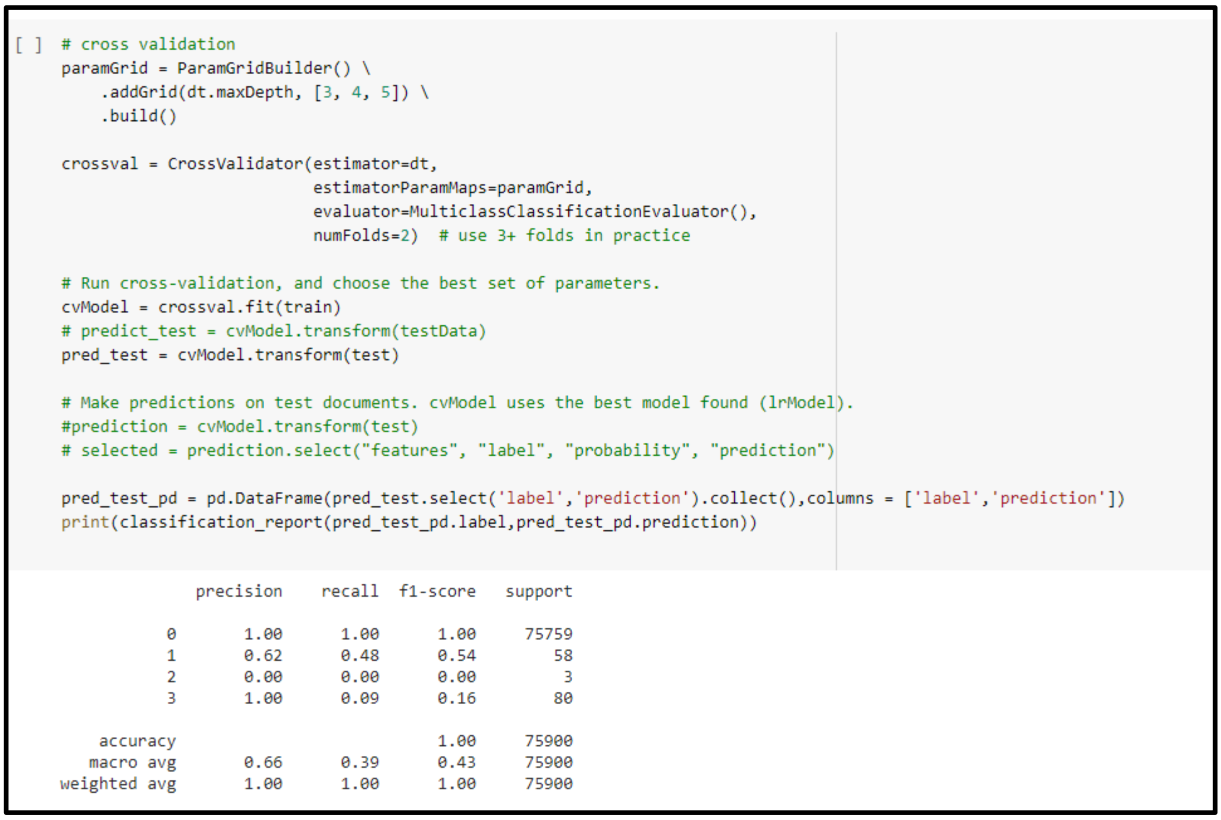
Decision Tree:



```
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
# Your code for 2-fold cross validation
```

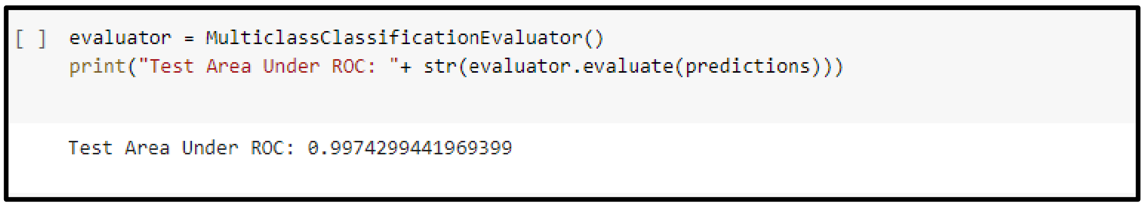
```
[ ] # decision tree
from pyspark.ml.classification import DecisionTreeClassifier
dt = DecisionTreeClassifier(featuresCol = 'features', labelCol = 'label', maxDepth = 3)
dtModel = dt.fit(train)
predictions = dtModel.transform(test)
predictions.select('features', 'label', 'rawPrediction', 'prediction', 'probability').show(10)
```

```
+--------------------+-----+--------------------+----------+--------------------+
|            features|label|       rawPrediction|prediction|         probability|
+--------------------+-----+--------------------+----------+--------------------+
|[4.0,2076.0,1734....|    0|[176710.0,136.0,1...|       0.0|[0.99799508655013...|
|[4.0,2082.0,908.0...|    0|[176710.0,136.0,1...|       0.0|[0.99799508655013...|
|[7.0,1108.0,199.0...|    0|[176710.0,136.0,1...|       0.0|[0.99799508655013...|
|[7.0,1109.0,199.0...|    0|[176710.0,136.0,1...|       0.0|[0.99799508655013...|
|[7.0,2034.0,4251....|    0|[176710.0,136.0,1...|       0.0|[0.99799508655013...|
|[7.0,2075.0,4251....|    0|[176710.0,136.0,1...|       0.0|[0.99799508655013...|
|[7.0,2099.0,4251....|    0|[176710.0,136.0,1...|       0.0|[0.99799508655013...|
|[15.0,2502.0,567....|    0|[176710.0,136.0,1...|       0.0|[0.99799508655013...|
|[17.0,1845.0,1734...|    0|[176710.0,136.0,1...|       0.0|[0.99799508655013...|
|[17.0,2080.0,1734...|    0|[176710.0,136.0,1...|       0.0|[0.99799508655013...|
+--------------------+-----+--------------------+----------+--------------------+
only showing top 10 rows
```



```
[ ] # cross validation
paramGrid = ParamGridBuilder() \
    .addGrid(dt.maxDepth, [3, 4, 5]) \
    .build()

crossval = CrossValidator(estimator=dt,
                          estimatorParamMaps=paramGrid,
                          evaluator=MulticlassClassificationEvaluator(),
                          numFolds=2)  # use 3+ folds in practice

# Run cross-validation, and choose the best set of parameters.
cvModel = crossval.fit(train)
# predict_test = cvModel.transform(testData)
pred_test = cvModel.transform(test)

# Make predictions on test documents. cvModel uses the best model found (lrModel).
#prediction = cvModel.transform(test)
# selected = prediction.select("features", "label", "probability", "prediction")

pred_test_pd = pd.DataFrame(pred_test.select('label','prediction').collect(),columns = ['label','prediction'])
print(classification_report(pred_test_pd.label,pred_test_pd.prediction))
```

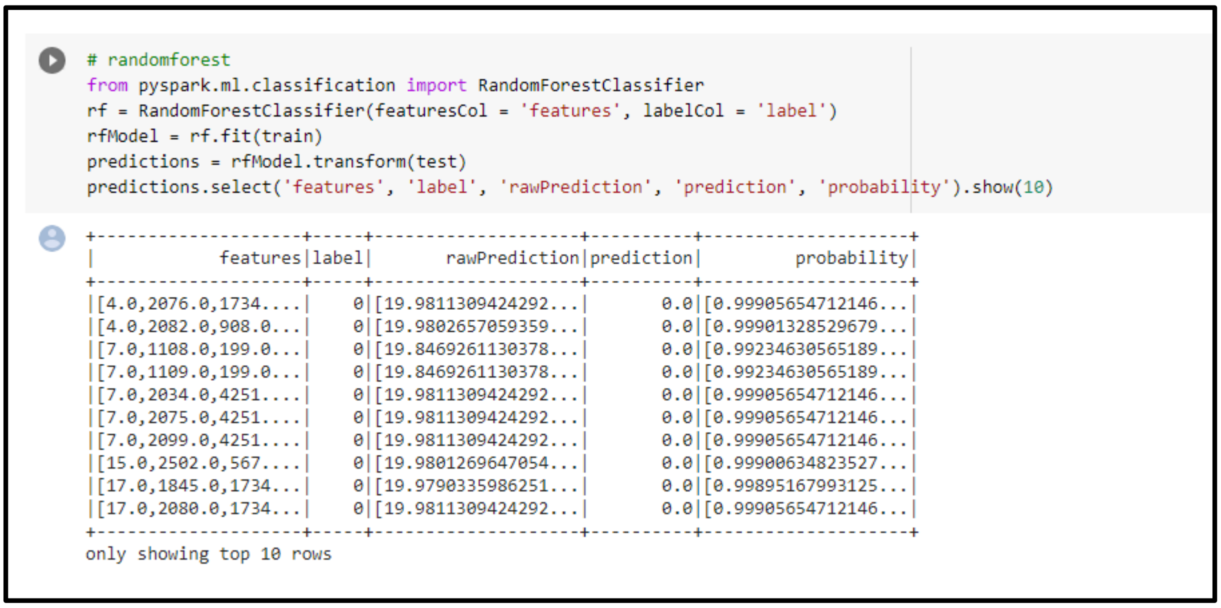
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     75759
           1       0.62      0.48      0.54        58
           2       0.00      0.00      0.00         3
           3       1.00      0.09      0.16        80

    accuracy                           1.00     75900
   macro avg       0.66      0.39      0.43     75900
weighted avg       1.00      1.00      1.00     75900
```



```
[ ] evaluator = MulticlassClassificationEvaluator()
    print("Test Area Under ROC: "+ str(evaluator.evaluate(predictions)))
```

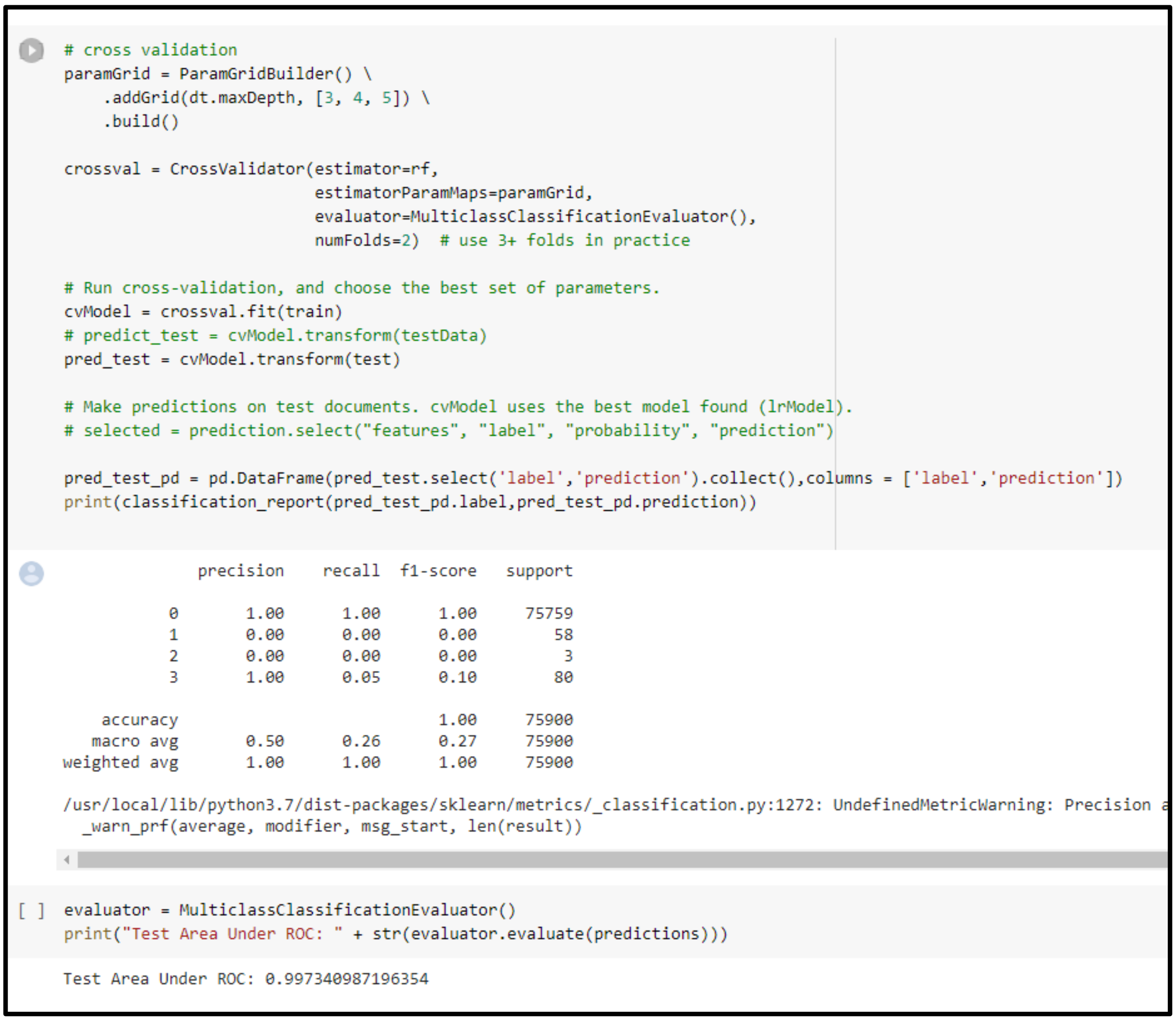
```
Test Area Under ROC: 0.9974299441969399
```

Random Forest:



```
# randomforest
from pyspark.ml.classification import RandomForestClassifier
rf = RandomForestClassifier(featuresCol = 'features', labelCol = 'label')
rfModel = rf.fit(train)
predictions = rfModel.transform(test)
predictions.select('features', 'label', 'rawPrediction', 'prediction', 'probability').show(10)
```

```
+--------------------+-----+--------------------+----------+--------------------+
|            features|label|       rawPrediction|prediction|         probability|
+--------------------+-----+--------------------+----------+--------------------+
|[4.0,2076.0,1734....|    0|[19.9811309424292...|       0.0|[0.99905654712146...|
|[4.0,2082.0,908.0...|    0|[19.9802657059359...|       0.0|[0.99901328529679...|
|[7.0,1108.0,199.0...|    0|[19.8469261130378...|       0.0|[0.99234630565189...|
|[7.0,1109.0,199.0...|    0|[19.8469261130378...|       0.0|[0.99234630565189...|
|[7.0,2034.0,4251....|    0|[19.9811309424292...|       0.0|[0.99905654712146...|
|[7.0,2075.0,4251....|    0|[19.9811309424292...|       0.0|[0.99905654712146...|
|[7.0,2099.0,4251....|    0|[19.9811309424292...|       0.0|[0.99905654712146...|
|[15.0,2502.0,567....|    0|[19.9801269647054...|       0.0|[0.99900634823527...|
|[17.0,1845.0,1734...|    0|[19.9790335986251...|       0.0|[0.99895167993125...|
|[17.0,2080.0,1734...|    0|[19.9811309424292...|       0.0|[0.99905654712146...|
+--------------------+-----+--------------------+----------+--------------------+
only showing top 10 rows
```



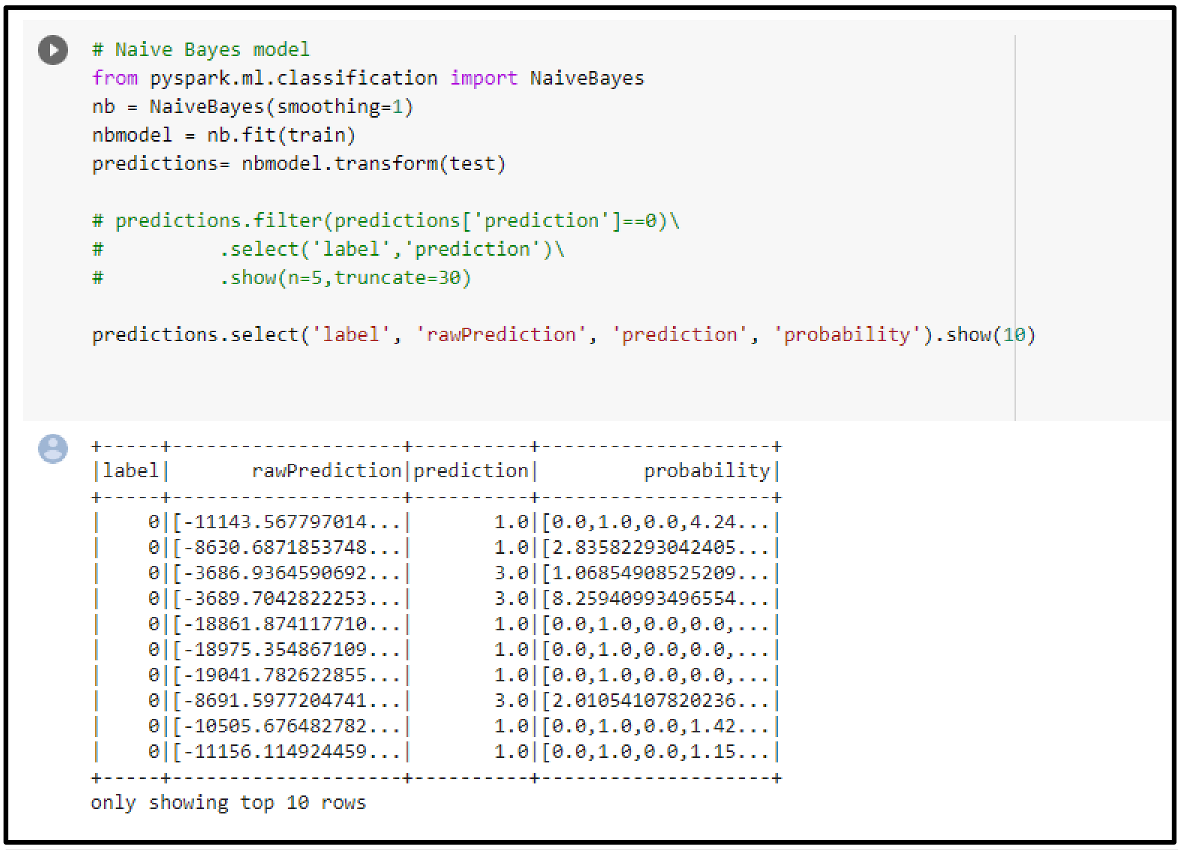
```
# cross validation
paramGrid = ParamGridBuilder() \
    .addGrid(dt.maxDepth, [3, 4, 5]) \
    .build()

crossval = CrossValidator(estimator=rf,
                          estimatorParamMaps=paramGrid,
                          evaluator=MulticlassClassificationEvaluator(),
                          numFolds=2)  # use 3+ folds in practice

# Run cross-validation, and choose the best set of parameters.
cvModel = crossval.fit(train)
# predict_test = cvModel.transform(testData)
pred_test = cvModel.transform(test)

# Make predictions on test documents. cvModel uses the best model found (lrModel).
# selected = prediction.select("features", "label", "probability", "prediction")

pred_test_pd = pd.DataFrame(pred_test.select('label','prediction').collect(),columns = ['label','prediction'])
print(classification_report(pred_test_pd.label,pred_test_pd.prediction))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     75759
           1       0.00      0.00      0.00        58
           2       0.00      0.00      0.00         3
           3       1.00      0.05      0.10        80

    accuracy                           1.00     75900
   macro avg       0.50      0.26      0.27     75900
weighted avg       1.00      1.00      1.00     75900

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

```
[ ] evaluator = MulticlassClassificationEvaluator()
    print("Test Area Under ROC: " + str(evaluator.evaluate(predictions)))
```

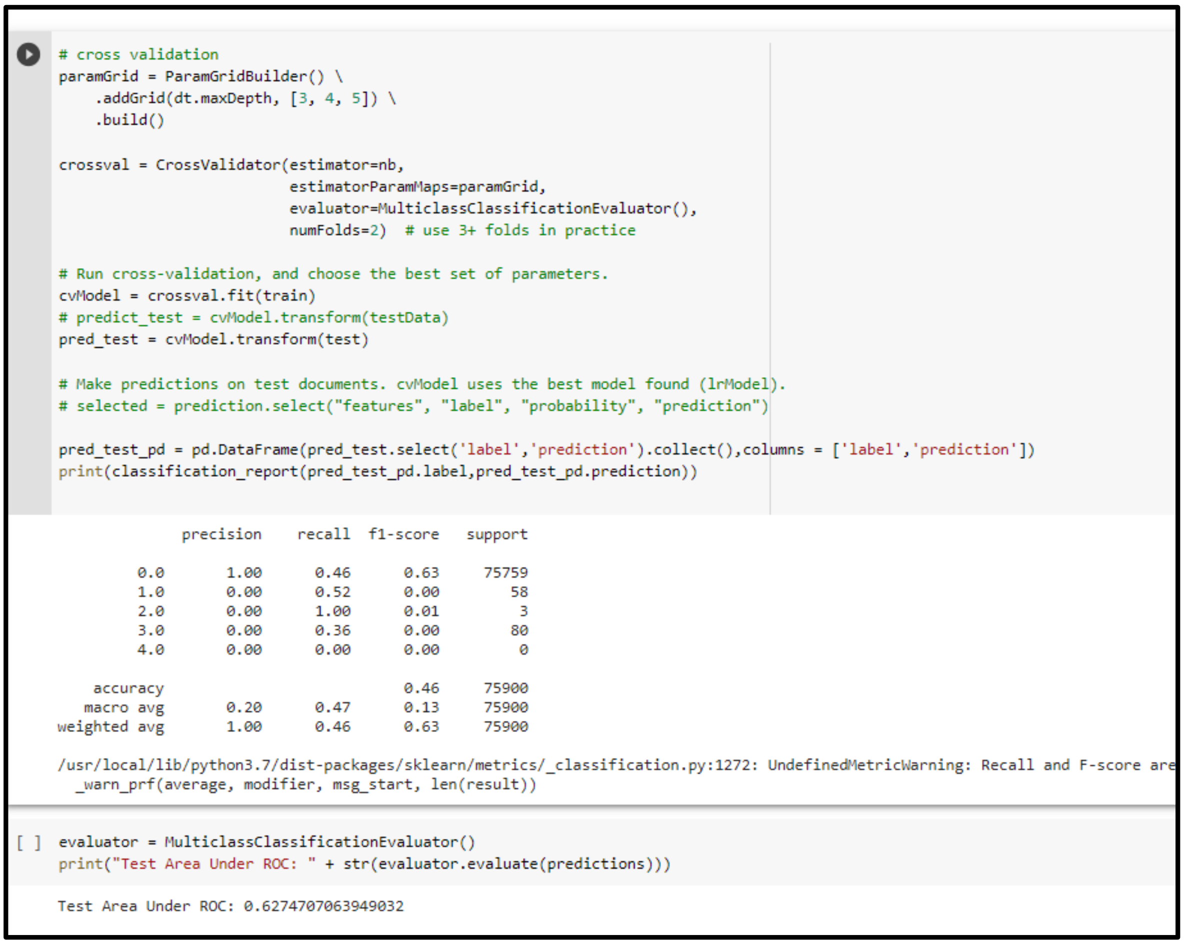
```
Test Area Under ROC: 0.997340987196354
```

Naïve Bayes:



```
# Naive Bayes model
from pyspark.ml.classification import NaiveBayes
nb = NaiveBayes(smoothing=1)
nbmodel = nb.fit(train)
predictions= nbmodel.transform(test)

# predictions.filter(predictions['prediction']==0)\
#          .select('label','prediction')\
#          .show(n=5,truncate=30)

predictions.select('label', 'rawPrediction', 'prediction', 'probability').show(10)
```

```
+-----+--------------------+----------+--------------------+
|label|       rawPrediction|prediction|         probability|
+-----+--------------------+----------+--------------------+
|    0|[-11143.567797014...|       1.0|[0.0,1.0,0.0,4.24...|
|    0|[-8630.6871853748...|       1.0|[2.83582293042405...|
|    0|[-3686.9364590692...|       3.0|[1.06854908525209...|
|    0|[-3689.7042822253...|       3.0|[8.25940993496554...|
|    0|[-18861.874117710...|       1.0|[0.0,1.0,0.0,0.0,...|
|    0|[-18975.354867109...|       1.0|[0.0,1.0,0.0,0.0,...|
|    0|[-19041.782622855...|       1.0|[0.0,1.0,0.0,0.0,...|
|    0|[-8691.5977204741...|       3.0|[2.01054107820236...|
|    0|[-10505.676482782...|       1.0|[0.0,1.0,0.0,1.42...|
|    0|[-11156.114924459...|       1.0|[0.0,1.0,0.0,1.15...|
+-----+--------------------+----------+--------------------+
only showing top 10 rows
```



```
# cross validation
paramGrid = ParamGridBuilder() \
    .addGrid(dt.maxDepth, [3, 4, 5]) \
    .build()

crossval = CrossValidator(estimator=nb,
                          estimatorParamMaps=paramGrid,
                          evaluator=MulticlassClassificationEvaluator(),
                          numFolds=2)  # use 3+ folds in practice

# Run cross-validation, and choose the best set of parameters.
cvModel = crossval.fit(train)
# predict_test = cvModel.transform(testData)
pred_test = cvModel.transform(test)

# Make predictions on test documents. cvModel uses the best model found (lrModel).
# selected = prediction.select("features", "label", "probability", "prediction")

pred_test_pd = pd.DataFrame(pred_test.select('label','prediction').collect(),columns = ['label','prediction'])
print(classification_report(pred_test_pd.label,pred_test_pd.prediction))
```

```
              precision    recall  f1-score   support

         0.0       1.00      0.46      0.63     75759
         1.0       0.00      0.52      0.00        58
         2.0       0.00      1.00      0.01         3
         3.0       0.00      0.36      0.00        80
         4.0       0.00      0.00      0.00         0

    accuracy                           0.46     75900
   macro avg       0.20      0.47      0.13     75900
weighted avg       1.00      0.46      0.63     75900

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Recall and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
```

```
[ ] evaluator = MulticlassClassificationEvaluator()
    print("Test Area Under ROC: " + str(evaluator.evaluate(predictions)))
```

```
Test Area Under ROC: 0.6274707063949032
```

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

Cross validation is a better way to deal with bias and variance. In this part, it makes k-fold as 2, it means cross validation splits data randomly into 2 folds, then k-1 part is used for model training and 1 part to evaluate it, meanwhile, the validation will run k times, which can ensure every data point can be used in the model training.

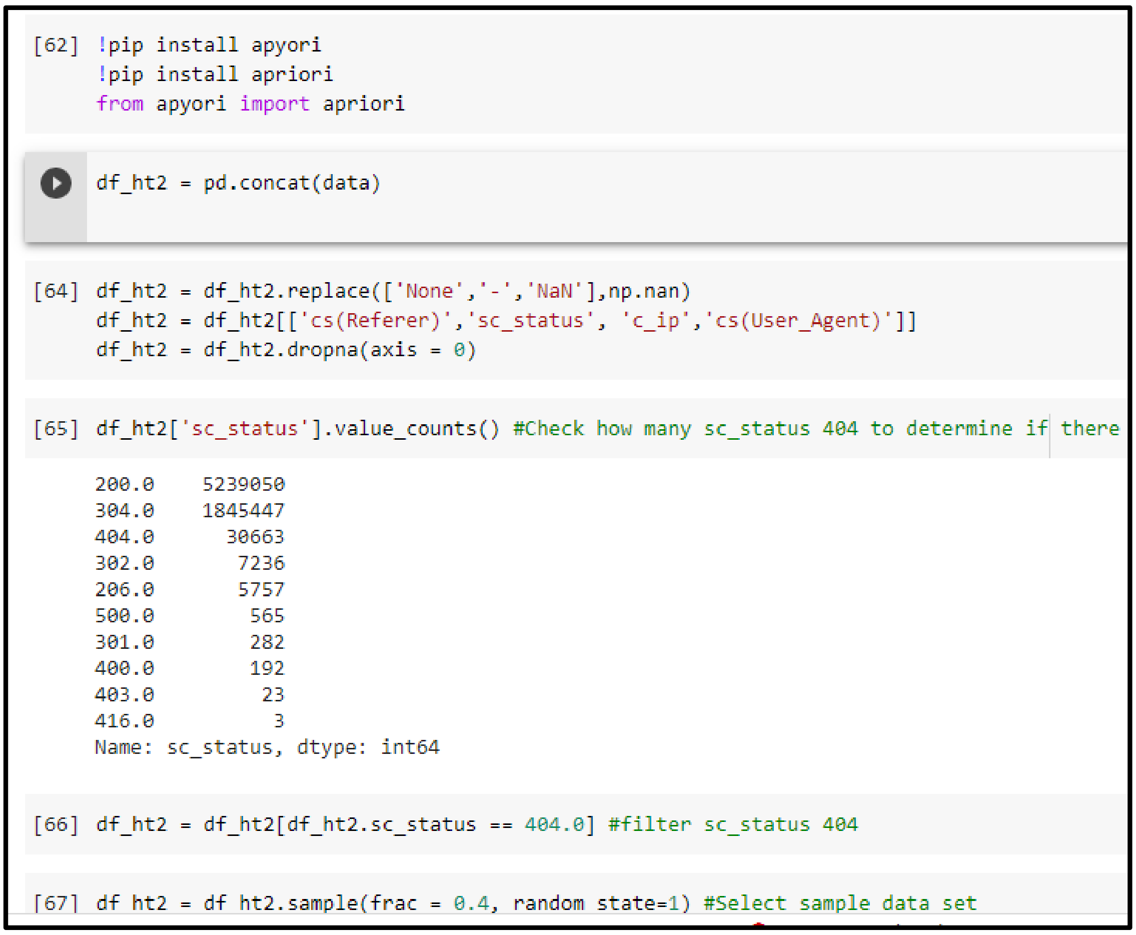
To determine which of the 3 validation models are the most accurate model, we need to look at the F1-scores in each model as this is a better indication for datasets that have inbalanced classes which is the case for the data set that we are using. The F1 score is a metric that is used to find an equal balance between precision and recall. The F1 score ranges from a scale of 1.0 to 0.0 with 1 being the best and 0.0 being the worst

Between the 3 models, both the Random Forest Model and Decision Tree Model scored a F1 score of 1, for the class labelled 0. Naïve Bayes did not score 1 for any of the classes, so out of the 3, Naïve Bayes performed the worst. So we need to look at the F1 scores for the other classes between Random Forest and Decision Tree. In the Random Forest Model, classes 1 and 2 scored a score of 0, where as for Decision Tree class 1, scored 0.54 and class 2 scored 0. And for the final class, class 3, Decision tree scored 0.16 compared to 0.1 in Random forest. By looking at the F1 scores amongst all classes for the 3 validation models, we can confidently say that the Decision Tree model is the best model to use.

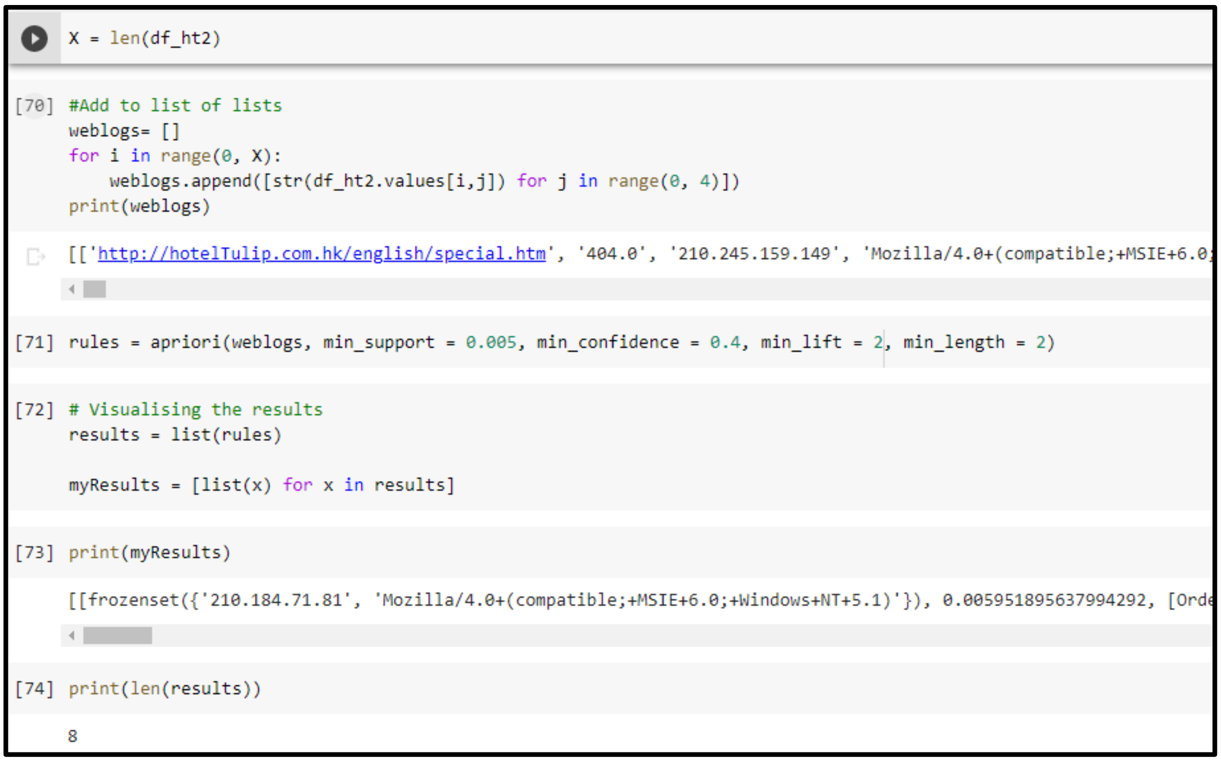
4. Association Rule Mining

Please add description of the following contents by yourselves.

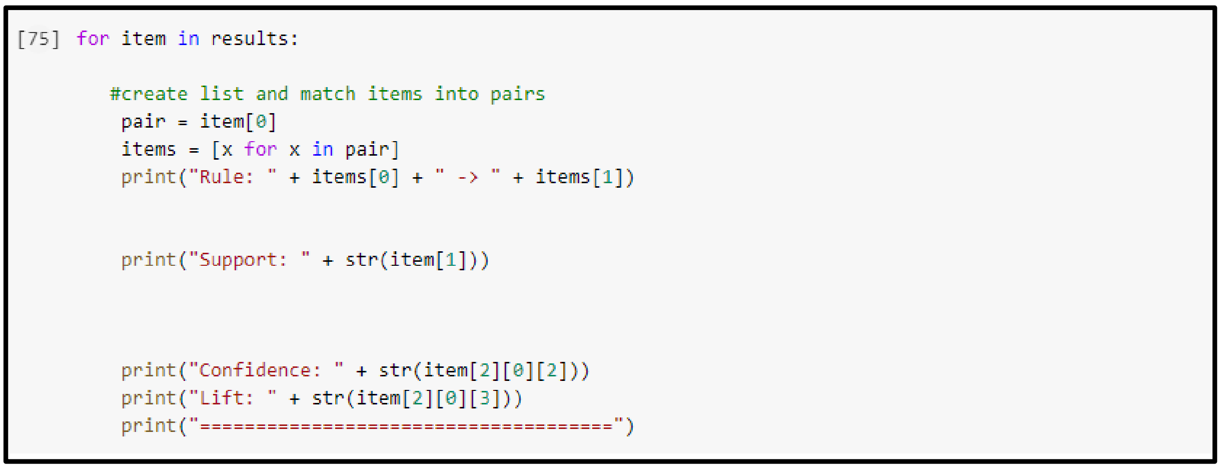
A. *Your code design and running results, threshold (parameter) for support and confidence*



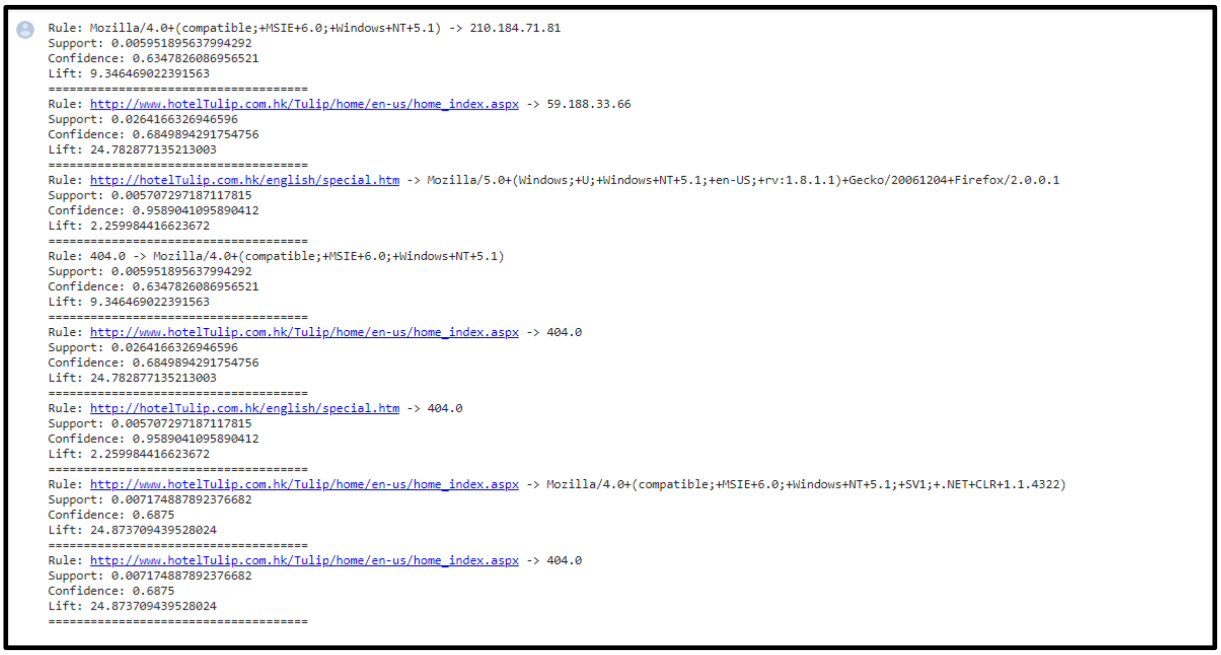
```
[62] !pip install apyori
     !pip install apriori
     from apyori import apriori

     df_ht2 = pd.concat(data)

[64] df_ht2 = df_ht2.replace(['None','-','NaN'],np.nan)
     df_ht2 = df_ht2[['cs(Referer)','sc_status', 'c_ip','cs(User_Agent)']]
     df_ht2 = df_ht2.dropna(axis = 0)

[65] df_ht2['sc_status'].value_counts() #Check how many sc_status 404 to determine if there

     200.0    5239050
     304.0    1845447
     404.0      30663
     302.0       7236
     206.0       5757
     500.0        565
     301.0        282
     400.0        192
     403.0         23
     416.0          3
     Name: sc_status, dtype: int64

[66] df_ht2 = df_ht2[df_ht2.sc_status == 404.0] #filter sc_status 404

[67] df_ht2 = df_ht2.sample(frac = 0.4, random_state=1) #Select sample data set
```



```
     X = len(df_ht2)

[70] #Add to list of lists
     weblogs= []
     for i in range(0, X):
         weblogs.append([str(df_ht2.values[i,j]) for j in range(0, 4)])
     print(weblogs)

     [['http://hotelTulip.com.hk/english/special.htm', '404.0', '210.245.159.149', 'Mozilla/4.0+(compatible;+MSIE+6.0;

[71] rules = apriori(weblogs, min_support = 0.005, min_confidence = 0.4, min_lift = 2, min_length = 2)

[72] # Visualising the results
     results = list(rules)

     myResults = [list(x) for x in results]

[73] print(myResults)

     [[frozenset({'210.184.71.81', 'Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1)'}), 0.005951895637994292, [Orde

[74] print(len(results))

     8
```



```
[75] for item in results:

        #create list and match items into pairs
         pair = item[0]
         items = [x for x in pair]
         print("Rule: " + items[0] + " -> " + items[1])


         print("Support: " + str(item[1]))



         print("Confidence: " + str(item[2][0][2]))
         print("Lift: " + str(item[2][0][3]))
         print("=====================================")
```



```
Rule: Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1) -> 210.184.71.81
Support: 0.005951895637994292
Confidence: 0.6347826086956521
Lift: 9.346469022391563
=====================================
Rule: http://www.hotelTulip.com.hk/Tulip/home/en-us/home_index.aspx -> 59.188.33.66
Support: 0.0264166326946596
Confidence: 0.6849894291754756
Lift: 24.782877135213003
=====================================
Rule: http://hotelTulip.com.hk/english/special.htm -> Mozilla/5.0+(Windows;+U;+Windows+NT+5.1;+en-US;+rv:1.8.1.1)+Gecko/20061204+Firefox/2.0.0.1
Support: 0.005707297187117815
Confidence: 0.9589041095890412
Lift: 2.259984416623672
=====================================
Rule: 404.0 -> Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1)
Support: 0.005951895637994292
Confidence: 0.6347826086956521
Lift: 9.346469022391563
=====================================
Rule: http://www.hotelTulip.com.hk/Tulip/home/en-us/home_index.aspx -> 404.0
Support: 0.0264166326946596
Confidence: 0.6849894291754756
Lift: 24.782877135213003
=====================================
Rule: http://hotelTulip.com.hk/english/special.htm -> 404.0
Support: 0.005707297187117815
Confidence: 0.9589041095890412
Lift: 2.259984416623672
=====================================
Rule: http://www.hotelTulip.com.hk/Tulip/home/en-us/home_index.aspx -> Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1;+.NET+CLR+1.1.4322)
Support: 0.007174887892376682
Confidence: 0.6875
Lift: 24.873709439528024
=====================================
Rule: http://www.hotelTulip.com.hk/Tulip/home/en-us/home_index.aspx -> 404.0
Support: 0.007174887892376682
Confidence: 0.6875
Lift: 24.873709439528024
=====================================
```

B. *Your findings on association rule mining results*

Apriori Algorithm is a mining technique used to identify patterns in a dataset and identify items or activities that have a strong association with each other. This mining technique can be used to identify trends in a dataset.

There are 3 metrics to measure associations. In this experiment here, support is measuring how frequent one of the other 3 attributes (Ip_address, browser type, and weblinks are seen with 404 error server status. Confidence is measuring the conditional probability that one of those 3 attributes are seen with the server status 404 and the final metric is lift which is the ratio of confidence to expected confidence, so essentially it is a measure of how strong the rule is.

Using the weblog dataset, we were able to generate 8 association rules for the server status of 404 page not found. A small sample set of about 12000 weblogs was used in the mining, a minimum support level of 0.5% was set with a minimum confidence level of 40%. The lift was set to 2 as lift greater than 1 is considered a strong association.

By running the data set through the model, we were able to understand if there are certain combinations of weblinks, types of browsers being used or ip addresses that are more prone to receiving this 404 server status. Of the 8 rules generated, there were 3 strong rules that were associated with the 404 server status, and they are:

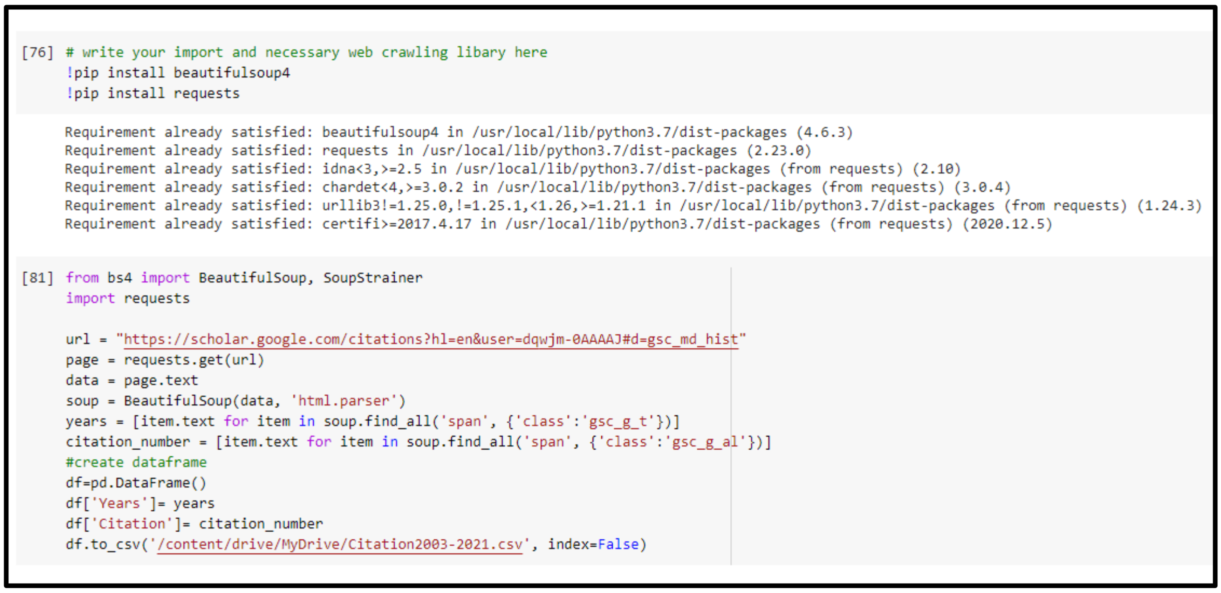
1) There was 2% of weblog requests that received the status of 404 while accessing the following weblink **"http://www.hotelTulip.com.hk/Tulip/home/en-us/home_index.aspx".** The confidence level sits at 68.5% which means 68.5% of the 404 errors are received through this weblink, and the lift for this rule is a strong 24.8% which is the conditional probability of someone accessing this weblink if they have received the 404 error. There is a very high likelihood that this weblink is a broken page.
2) There was 0.7% of weblog requests that received the status of 404 while accessing the following weblink **"http://www.hotelTulip.com.hk/Tulip/home/en-us/home_index.aspx"**. The confidence level sits at 68.5% which means 68.5% of the 404 errors are received through this weblink, and the lift for this rule is a strong 24.9% which is the conditional probability of someone accessing this weblink if they have received the 404 error. There is a very high likelihood that this weblink is a broken page.
3) There was 0.5% of weblog requests that received the status of 404 while accessing a weblink using the browser type of "**Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1)**". The confidence level sits at 63.4% which means 63.4% of the 404 errors are received through this browser, and the lift for this rule is a strong 9.3% which is the conditional probability of someone using this browser if they have received the 404 error. There is a very high likelihood that the browser used does not support the weblink that is being accessed.

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

Please fill this part with the screenshot of your code for generating citation2003-2021.csv.



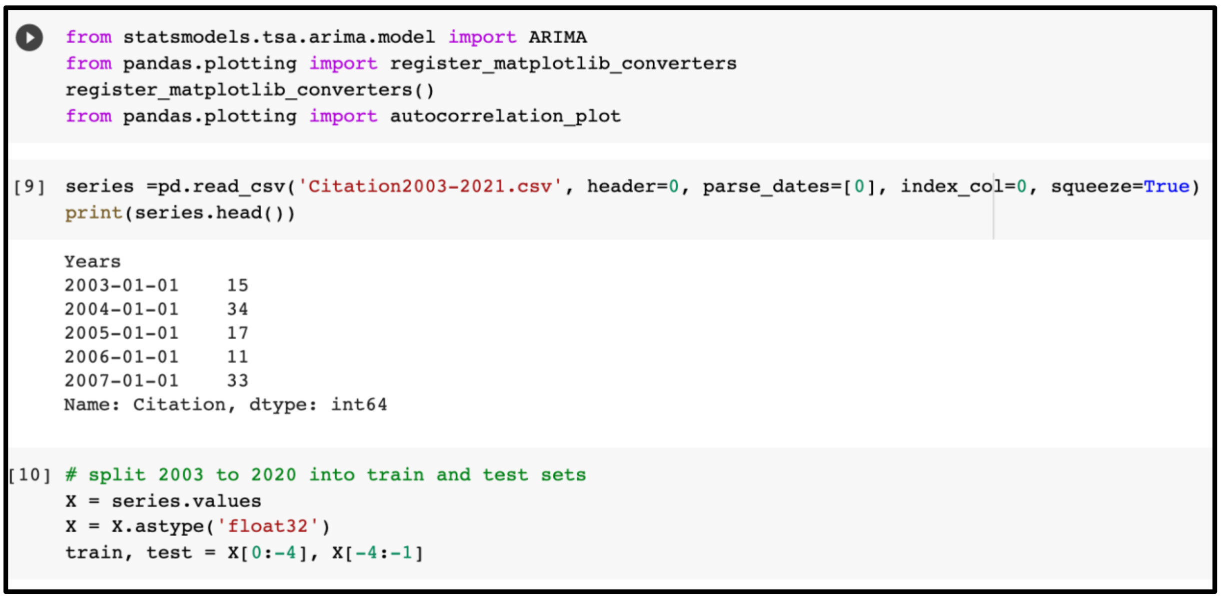
```
[76] # write your import and necessary web crawling libary here
     !pip install beautifulsoup4
     !pip install requests

     Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.7/dist-packages (4.6.3)
     Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (2.23.0)
     Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests) (2.10)
     Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests) (3.0.4)
     Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests) (1.24.3)
     Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests) (2020.12.5)
```

```
[81] from bs4 import BeautifulSoup, SoupStrainer
     import requests

     url = "https://scholar.google.com/citations?hl=en&user=dqwjm-0AAAAJ#d=gsc_md_hist"
     page = requests.get(url)
     data = page.text
     soup = BeautifulSoup(data, 'html.parser')
     years = [item.text for item in soup.find_all('span', {'class':'gsc_g_t'})]
     citation_number = [item.text for item in soup.find_all('span', {'class':'gsc_g_al'})]
     #create dataframe
     df=pd.DataFrame()
     df['Years']= years
     df['Citation']= citation_number
     df.to_csv('/content/drive/MyDrive/Citation2003-2021.csv', index=False)
```

6. Train Arima to predict the 2018 to 2020 citation

6.1. Train Arima Model

Please fill this part with the screenshot of your code for Arima Training.

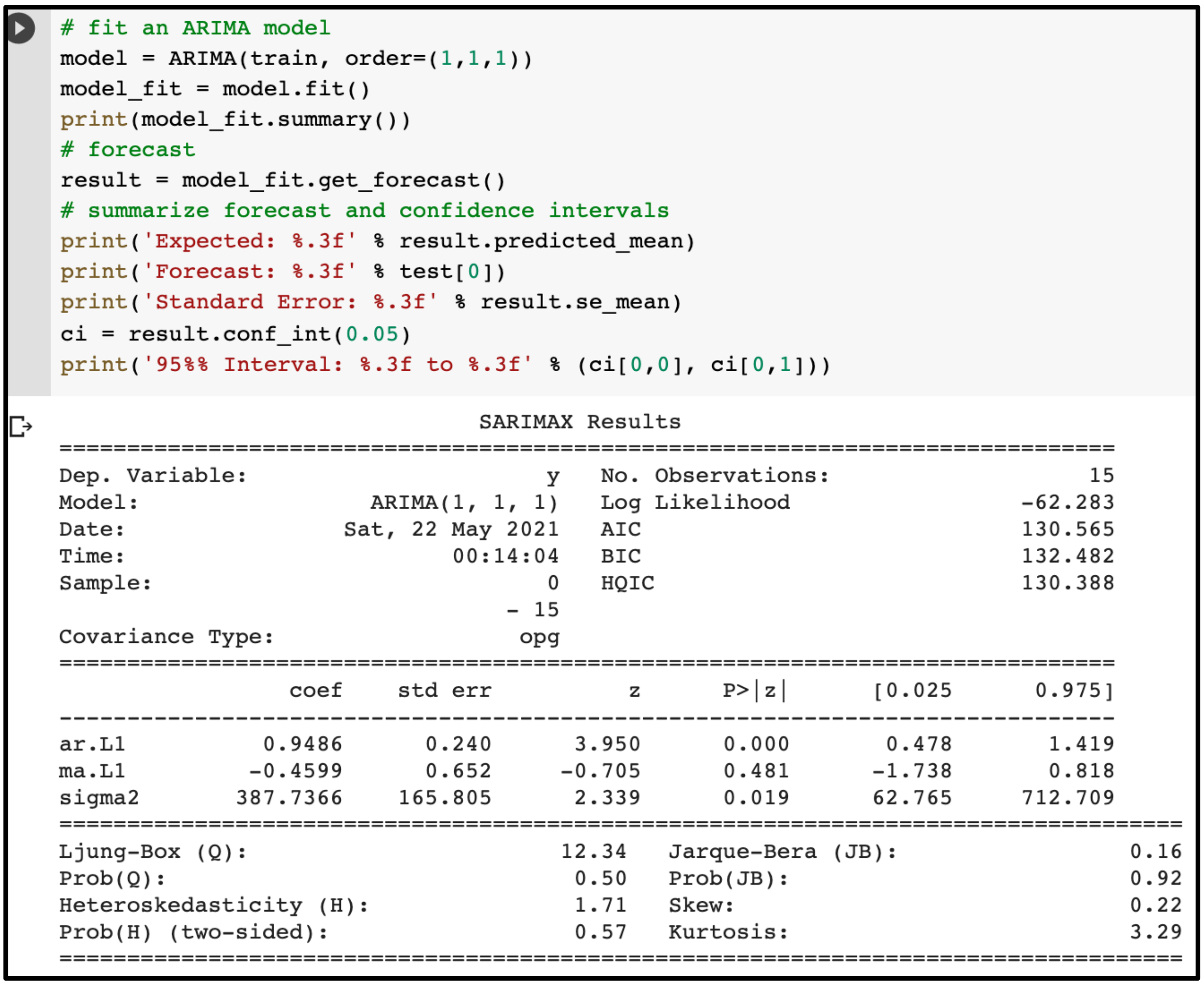


```
from statsmodels.tsa.arima.model import ARIMA
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
from pandas.plotting import autocorrelation_plot
```

```
[9]  series =pd.read_csv('Citation2003-2021.csv', header=0, parse_dates=[0], index_col=0, squeeze=True)
     print(series.head())
```

```
Years
2003-01-01    15
2004-01-01    34
2005-01-01    17
2006-01-01    11
2007-01-01    33
Name: Citation, dtype: int64
```

```
[10] # split 2003 to 2020 into train and test sets
     X = series.values
     X = X.astype('float32')
     train, test = X[0:-4], X[-4:-1]
```

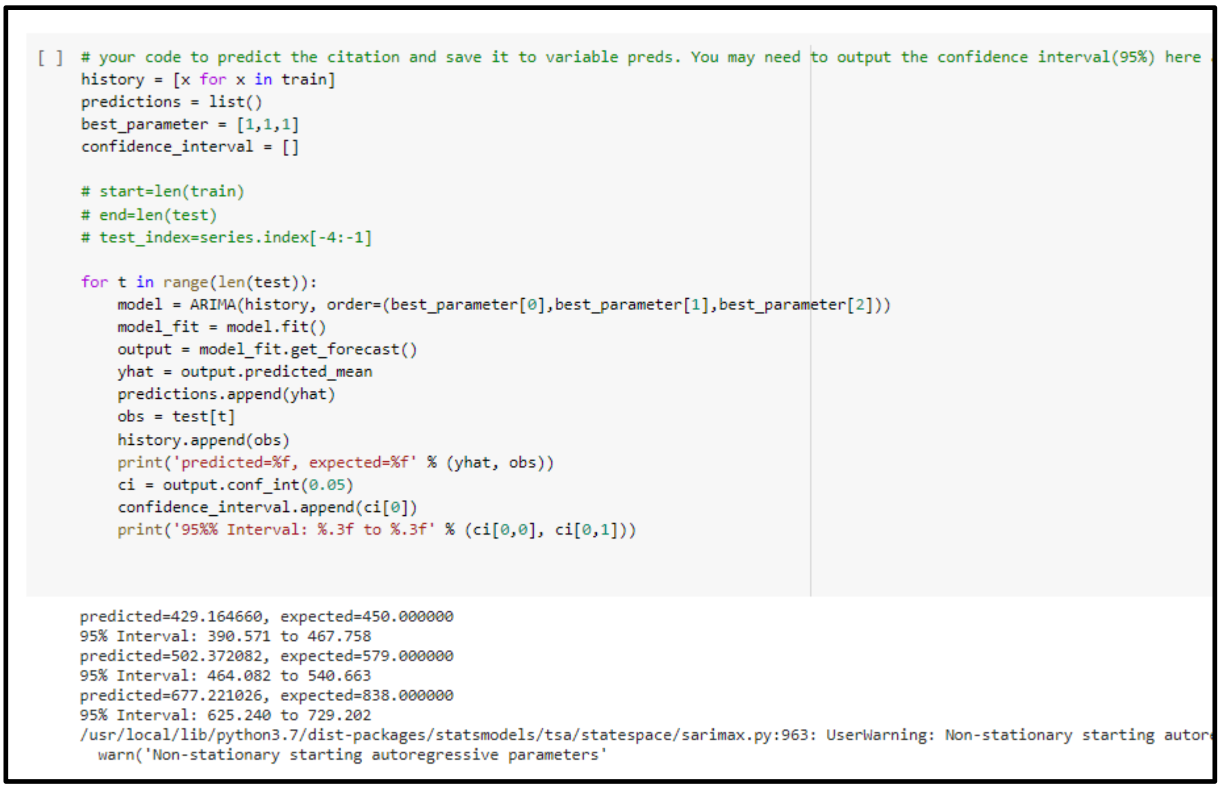


```
# fit an ARIMA model
model = ARIMA(train, order=(1,1,1))
model_fit = model.fit()
print(model_fit.summary())
# forecast
result = model_fit.get_forecast()
# summarize forecast and confidence intervals
print('Expected: %.3f' % result.predicted_mean)
print('Forecast: %.3f' % test[0])
print('Standard Error: %.3f' % result.se_mean)
ci = result.conf_int(0.05)
print('95%% Interval: %.3f to %.3f' % (ci[0,0], ci[0,1]))
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   15
Model:                 ARIMA(1, 1, 1)   Log Likelihood                 -62.283
Date:                Sat, 22 May 2021   AIC                            130.565
Time:                        00:14:04   BIC                            132.482
Sample:                             0   HQIC                           130.388
                                 - 15
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.9486      0.240      3.950      0.000       0.478       1.419
ma.L1         -0.4599      0.652     -0.705      0.481      -1.738       0.818
sigma2       387.7366    165.805      2.339      0.019      62.765     712.709
===================================================================================
Ljung-Box (Q):                       12.34   Jarque-Bera (JB):                 0.16
Prob(Q):                              0.50   Prob(JB):                         0.92
Heteroskedasticity (H):               1.71   Skew:                             0.22
Prob(H) (two-sided):                  0.57   Kurtosis:                         3.29
===================================================================================
```

6.2. Predicting the citation and Calculate the RMSE

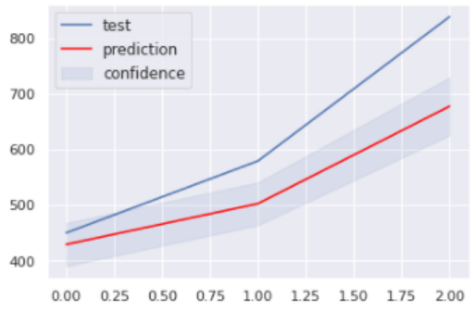
Please fill this part with the screenshot of your code for predicting and display RMSE (root mean square error).



```
[ ]  # your code to predict the citation and save it to variable preds. You may need to output the confidence interval(95%) here
     history = [x for x in train]
     predictions = list()
     best_parameter = [1,1,1]
     confidence_interval = []

     # start=len(train)
     # end=len(test)
     # test_index=series.index[-4:-1]

     for t in range(len(test)):
         model = ARIMA(history, order=(best_parameter[0],best_parameter[1],best_parameter[2]))
         model_fit = model.fit()
         output = model_fit.get_forecast()
         yhat = output.predicted_mean
         predictions.append(yhat)
         obs = test[t]
         history.append(obs)
         print('predicted=%f, expected=%f' % (yhat, obs))
         ci = output.conf_int(0.05)
         confidence_interval.append(ci[0])
         print('95%% Interval: %.3f to %.3f' % (ci[0,0], ci[0,1]))
```

```
     predicted=429.164660, expected=450.000000
     95% Interval: 390.571 to 467.758
     predicted=502.372082, expected=579.000000
     95% Interval: 464.082 to 540.663
     predicted=677.221026, expected=838.000000
     95% Interval: 625.240 to 729.202
     /usr/local/lib/python3.7/dist-packages/statsmodels/tsa/statespace/sarimax.py:963: UserWarning: Non-stationary starting autor
       warn('Non-stationary starting autoregressive parameters'
```

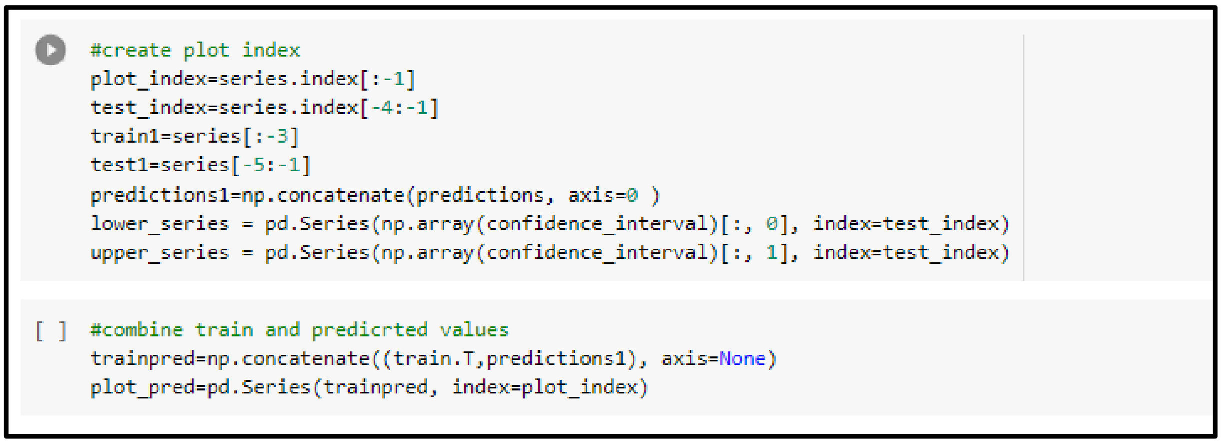
```
# plot forecasts against actual outcomes and also the confidence int at 95%
plt.plot(test,label='test')
plt.plot(predictions, color='red',label='prediction')
plt.fill_between(list(range(len(test))),
                 np.array(confidence_interval)[:,0], np.array(confidence_interval)[:,1],
                alpha=0.1, color='b',label='confidence')
plt.legend(loc='upper left', fontsize=12)
plt.show()
```



```
[ ]  from sklearn.metrics import mean_squared_error
     from math import sqrt

     # evaluate forecasts
     # Your code to show the performance RMSE
     rmse = sqrt(mean_squared_error(test, predictions))
     print('Test RMSE: %.3f' % rmse)

     Test RMSE: 103.531
```

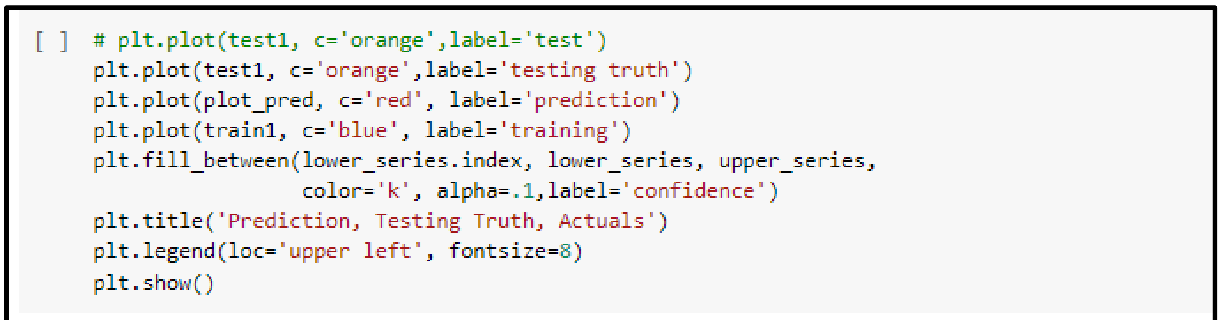
6.3. Draw the visualization to compare

Please fill this part with the screenshot of your line plot.

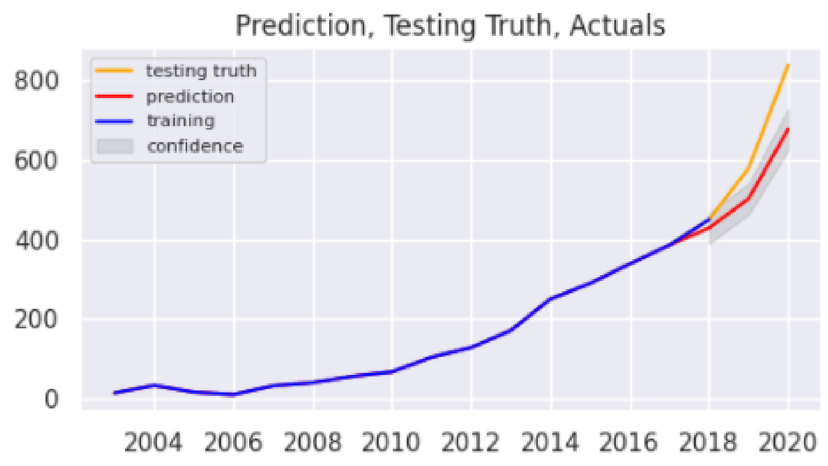


```
#create plot index
plot_index=series.index[:-1]
test_index=series.index[-4:-1]
train1=series[:-3]
test1=series[-5:-1]
predictions1=np.concatenate(predictions, axis=0 )
lower_series = pd.Series(np.array(confidence_interval)[:, 0], index=test_index)
upper_series = pd.Series(np.array(confidence_interval)[:, 1], index=test_index)
```

```
[ ] #combine train and predicrted values
    trainpred=np.concatenate((train.T,predictions1), axis=None)
    plot_pred=pd.Series(trainpred, index=plot_index)
```



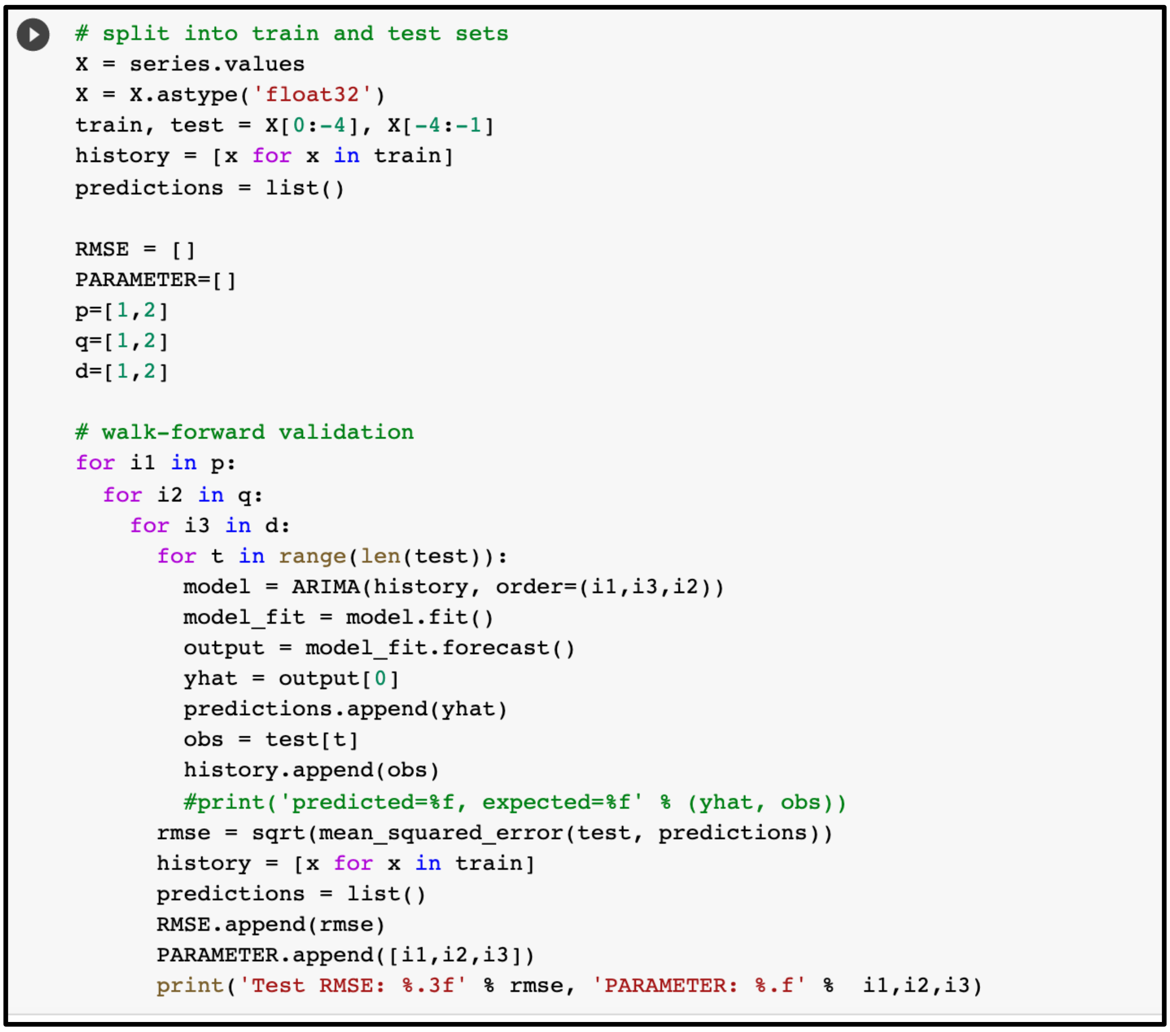
```
[ ] # plt.plot(test1, c='orange',label='test')
    plt.plot(test1, c='orange',label='testing truth')
    plt.plot(plot_pred, c='red', label='prediction')
    plt.plot(train1, c='blue', label='training')
    plt.fill_between(lower_series.index, lower_series, upper_series,
                     color='k', alpha=.1,label='confidence')
    plt.title('Prediction, Testing Truth, Actuals')
    plt.legend(loc='upper left', fontsize=8)
    plt.show()
```



7. Parameter selection and Year 2021 and 2022 Prediction

7.1. Grid Search

Please fill this part with the screenshot of your code for grid search.



```
# split into train and test sets
X = series.values
X = X.astype('float32')
train, test = X[0:-4], X[-4:-1]
history = [x for x in train]
predictions = list()

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

# walk-forward validation
for i1 in p:
  for i2 in q:
    for i3 in d:
      for t in range(len(test)):
        model = ARIMA(history, order=(i1,i3,i2))
        model_fit = model.fit()
        output = model_fit.forecast()
        yhat = output[0]
        predictions.append(yhat)
        obs = test[t]
        history.append(obs)
        #print('predicted=%f, expected=%f' % (yhat, obs))
      rmse = sqrt(mean_squared_error(test, predictions))
      history = [x for x in train]
      predictions = list()
      RMSE.append(rmse)
      PARAMETER.append([i1,i2,i3])
      print('Test RMSE: %.3f' % rmse, 'PARAMETER: %.f' %  i1,i2,i3)
```

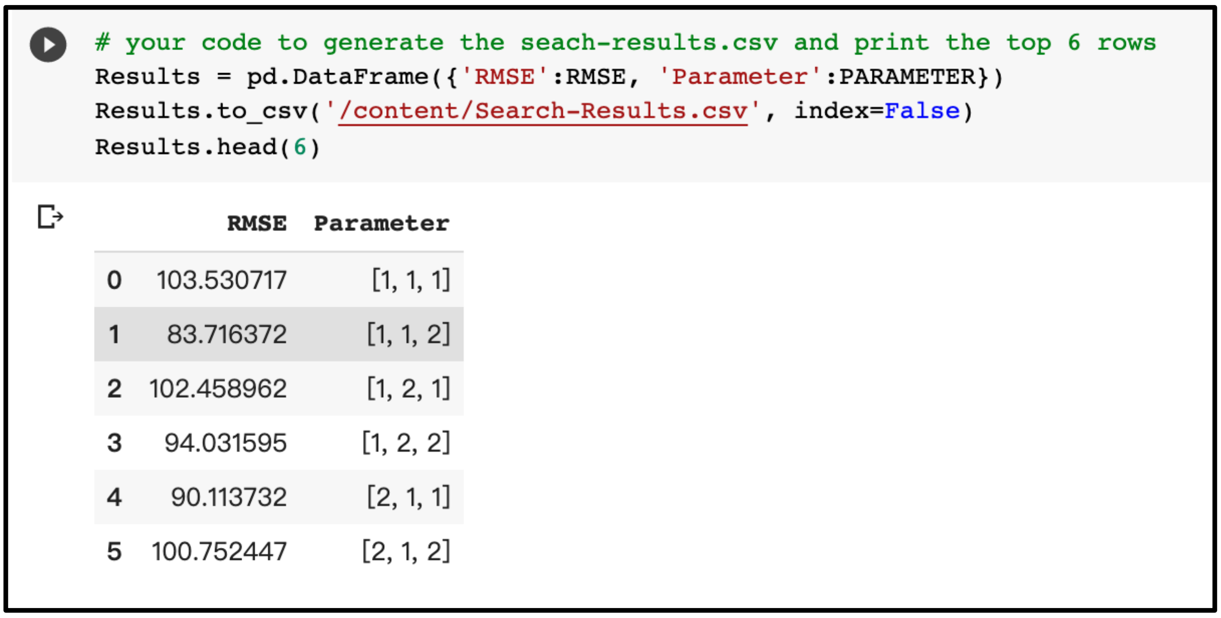
7.2. Select the best parameter values and Predict for 2021 and 2022

Please add descriptions of the following contents by yourselves.

A. Find out and display the best parameter values

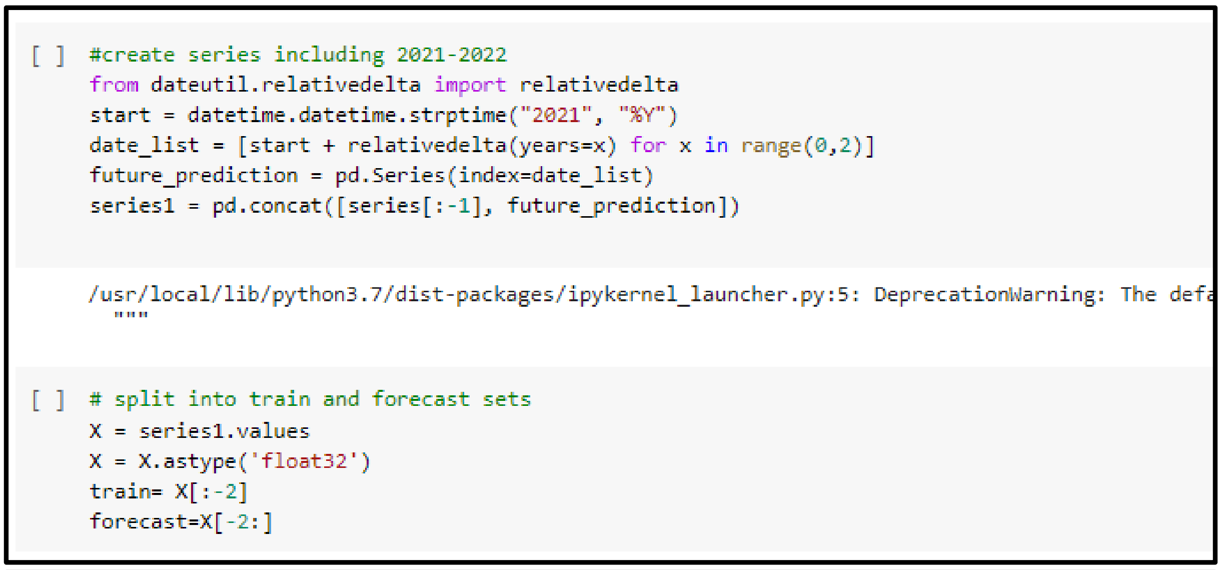
The RMSE is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data–how close the observed data points are to the model's predicted values. Whereas R-squared is a relative measure of fit, RMSE is an absolute measure of fit. As the square root of a variance, RMSE can be interpreted as the standard deviation of the unexplained variance, and has the useful property of being in the same units as the response variable. Lower values of RMSE indicate better fit. RMSE is a good measure of how accurately the model predicts the response, and it is the most important criterion for fit if the main purpose of the model is prediction. (Grace-Martin, n.d.)

In the case of our current model, the Parameter with the lowest RMSE is 1,1,2 with 83.7. therefore the best fit for our model.

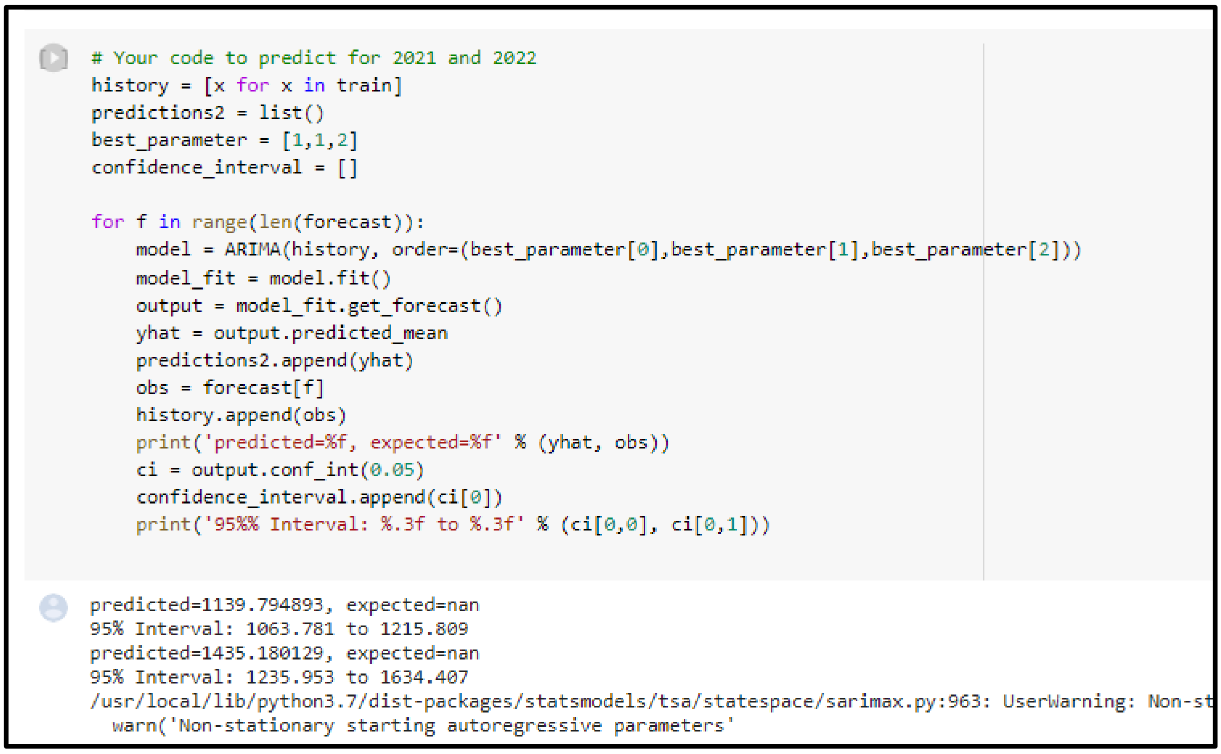


```
# your code to generate the seach-results.csv and print the top 6 rows
Results = pd.DataFrame({'RMSE':RMSE, 'Parameter':PARAMETER})
Results.to_csv('/content/Search-Results.csv', index=False)
Results.head(6)
```

	RMSE	Parameter
0	103.530717	[1, 1, 1]
1	83.716372	[1, 1, 2]
2	102.458962	[1, 2, 1]
3	94.031595	[1, 2, 2]
4	90.113732	[2, 1, 1]
5	100.752447	[2, 1, 2]



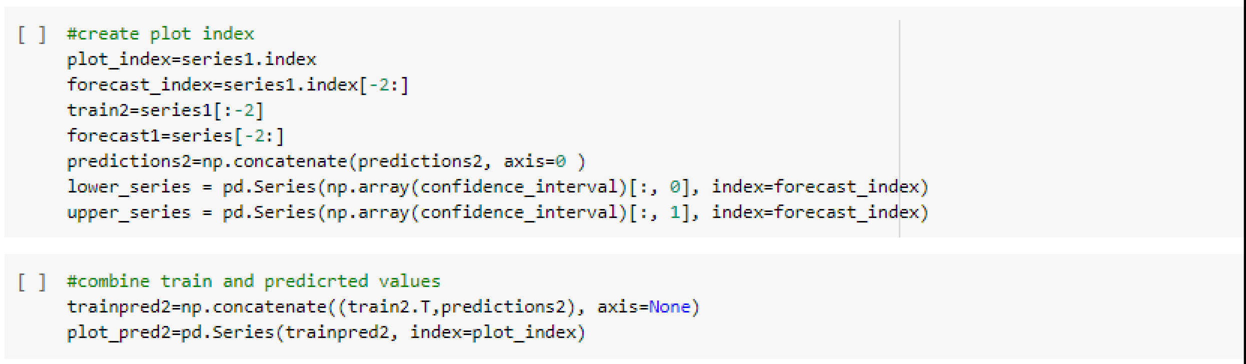
```
[ ] #create series including 2021-2022
    from dateutil.relativedelta import relativedelta
    start = datetime.datetime.strptime("2021", "%Y")
    date_list = [start + relativedelta(years=x) for x in range(0,2)]
    future_prediction = pd.Series(index=date_list)
    series1 = pd.concat([series[:-1], future_prediction])


    /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: DeprecationWarning: The defa
      """


[ ] # split into train and forecast sets
    X = series1.values
    X = X.astype('float32')
    train= X[:-2]
    forecast=X[-2:]
```



```
# Your code to predict for 2021 and 2022
history = [x for x in train]
predictions2 = list()
best_parameter = [1,1,2]
confidence_interval = []

for f in range(len(forecast)):
    model = ARIMA(history, order=(best_parameter[0],best_parameter[1],best_parameter[2]))
    model_fit = model.fit()
    output = model_fit.get_forecast()
    yhat = output.predicted_mean
    predictions2.append(yhat)
    obs = forecast[f]
    history.append(obs)
    print('predicted=%f, expected=%f' % (yhat, obs))
    ci = output.conf_int(0.05)
    confidence_interval.append(ci[0])
    print('95%% Interval: %.3f to %.3f' % (ci[0,0], ci[0,1]))
```

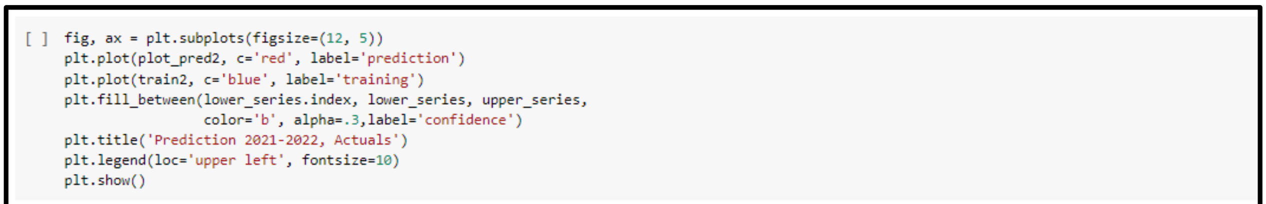
```
predicted=1139.794893, expected=nan
95% Interval: 1063.781 to 1215.809
predicted=1435.180129, expected=nan
95% Interval: 1235.953 to 1634.407
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/statespace/sarimax.py:963: UserWarning: Non-st
  warn('Non-stationary starting autoregressive parameters'
```

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

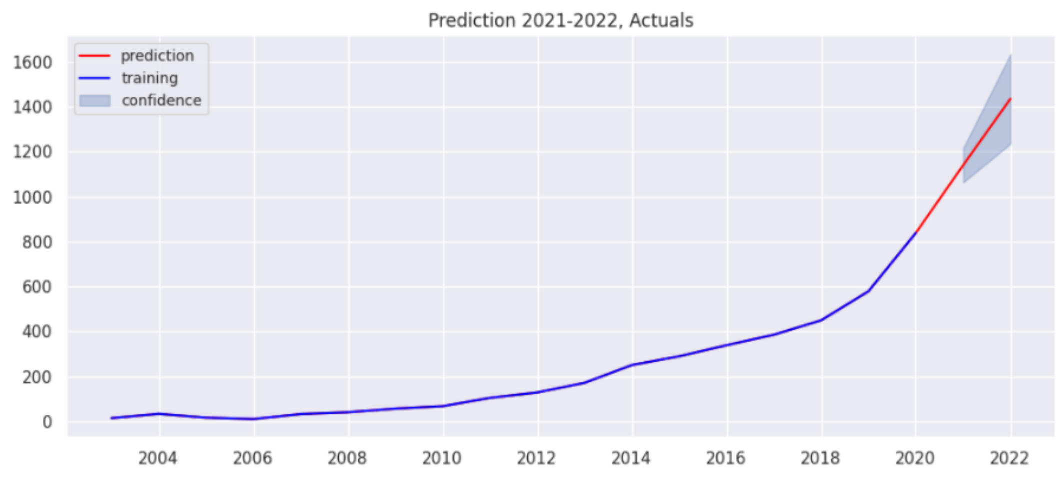


```
[ ] #create plot index
    plot_index=series1.index
    forecast_index=series1.index[-2:]
    train2=series1[:-2]
    forecast1=series[-2:]
    predictions2=np.concatenate(predictions2, axis=0 )
    lower_series = pd.Series(np.array(confidence_interval)[:, 0], index=forecast_index)
    upper_series = pd.Series(np.array(confidence_interval)[:, 1], index=forecast_index)
```

```
[ ] #combine train and predicrted values
    trainpred2=np.concatenate((train2.T,predictions2), axis=None)
    plot_pred2=pd.Series(trainpred2, index=plot_index)
```



```
[ ] fig, ax = plt.subplots(figsize=(12, 5))
    plt.plot(plot_pred2, c='red', label='prediction')
    plt.plot(train2, c='blue', label='training')
    plt.fill_between(lower_series.index, lower_series, upper_series,
                     color='b', alpha=.3,label='confidence')
    plt.title('Prediction 2021-2022, Actuals')
    plt.legend(loc='upper left', fontsize=10)
    plt.show()
```



Part III - Self Reflection - Essay

1) What are the Python packages that you find useful in manipulating and analyzing Big data? You can briefly analyze their advantages and disadvantages.

   Python is an extremely powerful platform to use when working with large datasets as it has many inbuilt libraries that are useful in performing data exploration and data wrangling tasks.

   Numpy:
   One of the key features of Numpy is its ability to perform complex mathematical calculations on multi-dimensional arrays. Python on its own does not support arrays but uses lists which does not support vectorized operations. Another advantage of using Numpy is that is utilises far less memory space and is much faster compared to regular python. (Duggal, 2021)

   Pandas:
   Efficient in completing data wrangling tasks such as removing missing values and duplicates, filtering and sorting, concatenating data sets, and performing grouping and aggregated calculations. (Duggal, 2021) It also has some inbuilt functions such as shape and describe which produces useful descriptive information for the data exploratory phase. Pandas is best suited to be used in the transformation stage of the ETL process. Pandas works best with structured tabular data sets as it utilises a 2-Dimensional DataFrame.

   Matplotlib & Seaborn:
   They are data visualisation libraries. Both these libraries produce graphs and plots that are useful in understanding the relationship between the variables in datasets. It also provides an object-oriented API, which can be used to embed those plots into applications (Duggal, 2021).

   Scikit-learn:
   A comprehensive machine learning library that provides supervised and unsupervised machine learning algorithms. The focus of the library is modelling data. Some popular models are: Clustering, cross validation, datasets, feature extraction/selection, dimensionality reduction amongst others (Brownlee, 2014). The scikit-learn library is very versatile and handy and serves real-world purposes. (Vadapalli, 2020)

BeautifulSoup:
A web crawling and data scraping library that helps in parsing HTML or XML documents into a tree structure to find and extract data. The library automatically converts incoming and outgoing documents to Unicode and UTF-8, respectively (Choudhury, 2020).

Scrapy:
For extracting data from websites in a fast and simple manner using APIs. It can also be used as a general-purpose web crawler. Scrapy is an application framework, which can be used for writing web spiders that crawl websites and extract data from them. (Choudhury, 2020)

Selenium:
This tool was developed for testing web applications. However, its use has far exceeded that as it can handle several automation tasks. Selenium is a browser-based web automation tool that can be used in various web browsers and supports multiple language binding libraries such as Python, Ruby, Java, etc (Prasanna, 2020).

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

Two of the most used big data packages with python is Spark and Hadoop. Hadoop is a framework that allows for the storage and processing of vast amount of data efficiently using a cluster of commodity hardware. The Hadoop Distributed File System (HDFS) is Hadoop's file storage module where it splits and makes copies of the data into several different nodes in a cluster thus it is highly fault tolerant. Map Reduce is Hadoop's processing module, that processes large volumes of data in separate nodes. The individual results are then aggregated to give the final output. YARN is Hadoop's application manager that manages the resources of the systems storing and processing the data in Hadoop. (Kalron, 2020)

Spark is an open-source distributed framework and a parallel engine that utilises in RAM memory for fast processing. Resilient Distributed Dataset (RDD) is the fundamental data structure of Spark which is used to perform faster and efficient MapReduce operations. like Hadoop, each dataset in a Spark RDD is divided into immutable partitions on different nodes in a cluster through the Spark Transformation API. RDD supports in-memory processing which is much faster than processing on a disk or network. RDD is used to run iterative algorithms, and it allows for the use of interactive data mining tools. (Diep, 2020)

Although Spark can run fast as it utilises in memory processing, it requires a large amount of memory as the process is loaded and kept in the memory space for caching purposes, compared to Hadoop's Map Reduce, where the process is killed once the task is completed, and no caching is stored. Therefore, Spark is suited for iterative and repetitive tasks and Hadoop is more suited for one off data transformation tasks. Another advantage of spark compared to Hadoop is Spark is far easier to program as it supports a variety of languages such as Scala, R, Java and Python, and it allows for streaming, batch processing and machine learning all in the same cluster. Hadoop is far more complex to program and requires complex lines of code. Due to Hadoop's limitation around batch processing, Spark is also the better option for iterative machine learning tasks utilising its in built MLLib library that includes regression and classification modules and can build machine learning pipelines. (Samuel, 2021)

Couple of disadvantages to note with both these platforms is that they are not suited to work with small data sets, and it lacks real time processing, as both platform work in batches, although Spark's streaming ability can support near real time processing as the batches are processed in micro batches. (Samuel, 2021)

3) Compare and contract the Python data analytical packages and their Spark packages.

   Python and Spark operate in a similar manner, however there are a few differences between the packages. Both Python and Spark can create data frames. Pandas data frame is considered more user-friendly option compared to a spark data frame. With Pandas data frames, there is a flexibility to wrangle data and transform certain rows and columns, Spark however does not support indexing and is designed to scale by the number of rows not columns and data frames are also immutable. (Kupferschmidt, Spark vs Pandas, part 2 - Spark, 2020)

   Both python and spark come with their own machine learning libraries. Scikit-Learn is a popular python option and MLlib is Spark's library. Scikit-Learn is a flexible and easy to use tool when performing machine learning tasks when the data set fits into the memory size, however when working with larger data sets, MLLib is more suitable as it runs off a distributed framework. Given that data visualisation is a key part of the machine learning process, Scikit-Learn is the better option between the two as it can be used alongside pandas and Matplotlib. (Johnson, 2019)

   Performing machine learning tasks in MLlib is a lengthier process compared to Scikit-Learn as with Scikit-Learn you can utilise the entire data frame in the algorithm, however with MLlib, all features need to be compressed into a single vector. (Johnson, 2019)

Scikit-Learn also has more inbuilt algorithms such as Random Forests, Nearest Neighbours Regression and K-means, compared to MLlib. It is also worth noting that Scikit-Learn is well documented thus more user friendly.

The language between Python and Spark is different as Python runs on an interpreted language model and Spark runs on a compiled language model what this means is that Python's code is executed immediately, and no build or compile step is required which is the case with Spark. This is an advantage of python as issues with the code can be detected much earlier on rather than towards the end of the project. (Kupferschmidt, Spark vs Pandas, part 3 - Scala vs Python, 2020)

Python has become the most preferred data analysis and machine learning tool due to its active community with a vast selection of libraries and resources. Pandas is a great tool to analyse small datasets on a single machine. However, when managing big datasets operations become very slow and difficult to handle with a Pandas dataframe, which does not support parallelization. Spark dataframes are excellent to build a scalable application as they are distributed on clusters. Spark is suitable for machine learning algorithms, as it allows programs to load and query data repeatedly. (Gupta, 2020)

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

With all the data being collected everywhere in this Big Data era that we are living in, two privacy risks that should be considered around discrimination and anonymity. As technology advances especially with the rise of internet of things devices, data that is being collected now paints a good picture of an individual's lifestyle and this makes it more challenging to anonymize the data. For example, almost everything that a person does online can be traced back to their location via an IP address or even through geo location data that is readily available on social media, with that information agencies or cyber criminals can use publicly available data sources alongside the data that is collected online to de-identify the data and pinpoint to a person's identity. This can lead to risks of identity theft or unethical targeting. (Charith Perera, 2015)

The issues of discrimination can arise as society's reliance on machine learning algorithms to make predictions based on patterns in a dataset increases. This can be seen as a risk in the healthcare and insurance sector. For example, an insurance company can build a model to predict based on the available healthcare records, who is at a higher risk at contracting certain illnesses and use those predictions to decide on what level of cover an individual should receive. An individual can be pushed to a higher premium level or denied access to cover due to an incorrect prediction even though they are perfectly healthy. These levels of discrimination can be unintentional or intentional, with intentional discrimination utilising data mining technologies being extremely difficult to prove. (Brian d'Alessandro, 2019)

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

   To address the discrimination issue when utilising predictive analytics in providing support or services such as social security support, healthcare support and insurance benefits, one must consider the data that is being used in the datasets that is fed into these predictive models. One needs to question types of data that can cause conscious or unconscious biasness such as gender, age and racial background. We need to ask, does this piece of information really need to be in the models to produce the desired outcome?

   Data professionals should also consider including a discrimination-aware auditing process when developing machine learning classification as discrimination can arise from misclassification of data that is used in developing machine learning Algorithms. The discrimination-aware auditing consists of 3 separate tests, where the first test is to check against the chosen discrimination metric that determines if the data discriminates against an attribute with respect to the target variable. The second test is to test and document attributes that are highly correlated with the chosen attribute and the final test is to perform some tests to find groups of attributes that are prone to statistical errors. If any of these 3 test scores below the set threshold level, the attributes that are most likely to cause discrimination in the model should be removed. By including these tests, data professionals can include ethical and legal considerations into the model that is being developed. (Brian d'Alessandro, 2019)

   With the risk of anonymity increasing as technology advances, organisations that are collecting vast amounts of data should in the first instance advice the user of the collection of data and the purpose of the data. Secondly, be utilising masking and de-identification techniques such as replacing sensitive attributes in a dataset with artificial data, data swapping where certain attributes are swapped with another record and hashing where one way encryption is placed on extremely sensitive data and the encryption cannot be reversed. (Olivia Angiuli, 2015)

   Another trend that is slowly arising is a decentralised search engine which gives online freedom whilst maintaining user’s privacy. Current Peer to Peer(P2P) search engines are slower and their accuracy can be questionable compared to centralised search engines such as Google. However, this issue can be resolved if more people join the network (Bender, Michel, Triantafillou, Weikum, & Zimmer, 2006).

As an example, we have YaCy, a P2P search engine. The peers' data is kept in a custom-written NoSQL database. YaCy respects user privacy. All password- or cookies-protected pages are excluded from indexing. Additionally, pages loaded using GET or POST parameters are not indexed by default. Thus, only publicly accessible, non-password-protected pages will be indexed (Yacy, 2021).

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

   As technology advances, data science is going to be an ever-evolving area. Being a professional in the data science field, it is crucial for one to stay on top and learn new tools that will support data science projects. This Modern Data Science unit has covered a wide array of fundamental concepts that is required in the different areas of data science. Overall, it has been a good experience learning these fundamental concepts, which will now help us get a deeper understanding of the field of data science.

References

Bender, M., Michel, S., Triantafillou, P., Weikum, G., & Zimmer, C. (2006). P2P content search: Give the web back to the people. *IPTPS*.

Brian d'Alessandro, C. O. (2019). Conscientious Classification: A Data Scientist's Guide to Discrimination-Aware Classification. 30.

Brownlee, J. (2014). *A Gentle Introduction to Scikit-Learn: A Python Machine Learning Library* . Retrieved 05 16, 2021, from https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/

Charith Perera, R. R. (2015). Big Data Privacy in the Internet of Things Era.

Choudhury, A. (2020). *Scrapy VS Beautiful Soup: A Comparison Of Web Crawling Tools*. Retrieved 05 16, 2021, from https://analyticsindiamag.com/scrapy-vs-beautiful-soup-a-comparison-of-web-crawling-tools/

Diep, N. (2020). *Big Data Analytics: Apache Spark vs Apache Hadoop*. Retrieved May 15, 2021, from https://towardsdatascience.com/big-data-analytics-apache-spark-vs-apache-hadoop-7cb77a7a9424

Duggal, N. (2021). *Top 10 Python Libraries For Data Science for 2021*. Retrieved 05 16, 2021, from https://www.simplilearn.com/top-python-libraries-for-data-science-article

Grace-Martin, K. (n.d.). *Assessing the Fit of Regression Models*. Retrieved from theanalysisfactor.com: https://www.theanalysisfactor.com/assessing-the-fit-of-regression-models/

Gupta, A. (2020). *A journey from Pandas to Spark Data Frames*. Retrieved 05 17, 2021, from https://www.indellient.com/blog/a-journey-from-pandas-to-spark-data-frames/

Johnson, S. (2019). *From scikit-learn to Spark ML*. Retrieved May 15, 2021, from https://towardsdatascience.com/from-scikit-learn-to-spark-ml-f2886fb46852

Kalron, A. (2020). *How do Hadoop and Spark Stack Up?* Retrieved May 13, 2021, from https://logz.io/blog/hadoop-vs-spark/#:~:text=Spark%20has%20been%20found%20to,Naive%20Bayes%20and%20k%2Dmeans.

Kupferschmidt, K. (2020). *Spark vs Pandas, part 2 - Spark*. Retrieved May 15, 2021, from https://towardsdatascience.com/spark-vs-pandas-part-2-spark-c57f8ea3a781

Kupferschmidt, K. (2020). *Spark vs Pandas, part 3 - Scala vs Python*. Retrieved May 15, 2021, from https://towardsdatascience.com/spark-vs-pandas-part-2-spark-c57f8ea3a781

Olivia Angiuli, J. B. (2015). *How to De-identify Your Data*. Retrieved May 15, 2021, from https://queue.acm.org/detail.cfm?id=2838930

Prasanna. (2020). *Explain the use of Selenium Webdriver with Python*. Retrieved 05 16, 2021, from https://prasannajyothi.medium.com/explain-the-use-of-selenium-webdriver-with-python-d7fc8e2d66d3

Samuel, N. (2021). *Hadoop MapReduce vs Spark: A Comprehensive Analysis*. Retrieved May 15, 2021, from https://hevodata.com/learn/mapreduce-vs-spark-a-comprehensive-analysis/

Vadapalli, P. (2020). *Scikit-learn in Python: Features, Prerequisites, Pros & Cons* . Retrieved 05 16, 2021, from https://www.upgrad.com/blog/scikit-learn-in-python/

Yacy. (2021). *FAQ - Frequently Asked Questions* . Retrieved 05 17, 2021, from https://yacy.net/faq/