

CSIP5203: Big Data Analytics Applications

Assignment 2: Big Data Analytics

P-Number ONLY



Table of Contents

Task 1: Analysing Data with Spark SQL	3
Introduction of dataset	3
Data Analysis	3
Discussion/Findings	13
Task 2: Implement machine learning algorithms for data analytics	13
Introduction of Problem	13
Approaches	13
Results Analysis	14
Discussion and conclusions	20
References and Bibliography	22
Appendix (if necessary)	24

Task 1: Analysing Data with Spark SQL

Introduction of dataset

The research paper has discussed police open source data, which is accessible at the <https://data.police.uk/data/> is the dataset, which is chosen for the analysis. The dataset is appropriate for the white range of the analytical use change it offers through the information on crimes reported in Different cities. Cumbria, Nottingham, Leicester are the main Four cities for the investigation each of which represent a distinct geographical region.

Data Analysis

The research paper will use Apache Spark SQL to carry out the analysis. One of the components of Apache spark, Spark SQL or first a programming interface for manipulating data with SQL queries. The research will include Trends over time, within cities and between cities, in addition to the frequency of different kinds of crime in certain areas (Fernando *et al.* 2023).

Important package and generating Spark data frame object

1.1 Import Package

```
from csv import reader
from pyspark.sql import Row
from pyspark.sql import SparkSession
from pyspark.sql.types import *
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
import warnings

import os
os.environ["PYSPARK_PYTHON"] = "python3"

[ ] # Import data from SF government official website
import urllib.request
urllib.request.urlretrieve("https://data.sfgov.org/api/views/tmnf-yvry/rows.csv?accessType=DOWNLOAD", "/tmp/myxxxx.csv")
dbutils.fs.mv("file:/tmp/myxxxx.csv", "dbfs:/lailoffer/spark_hw1/data/sf_03_18.csv")
display(dbutils.fs.ls("dbfs:/lailoffer/spark_hw1/data/"))
```

The data set is uploaded and generated by Spark data frame object by using above code. Output has shown in below section:

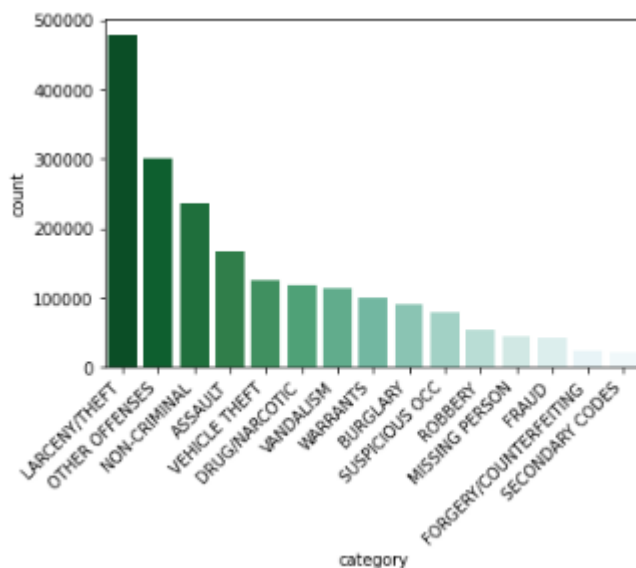
PdId	IncidentNum	Incident Code	Category	Descript	DayOfWeek	Date	Time	PdDistrict	Resolution	...	Fix It Zones as of 2017-11-06 2 2	DELETE - HSOC Zones as of 2018-02-07 2 2	Fix It Zones as of 2018-02-07 2 2	CBD, BID and GBD Boundaries as of 2017 2 2	Areas of Vulnerability, 2016 2 2	Market/Ter Bounc
0	4133422003074	041334220	03074	ROBBERY, BODILY FORCE	Monday	11/22/2004	17:50	INGLESIDE	NONE	...	None	None	None	None	None	
1	5118535807021	051185358	07021	VEHICLE THEFT, STOLEN AUTOMOBILE	Tuesday	10/18/2005	20:00	PARK	NONE	...	None	None	None	None	None	
2	4018830907021	040188309	07021	VEHICLE THEFT, STOLEN AUTOMOBILE	Sunday	02/15/2004	02:00	SOUTHERN	NONE	...	None	None	None	None	None	
3	11014543126030	110145431	26030	ARSON	Friday	02/18/2011	05:27	INGLESIDE	NONE	...	None	None	None	None	1	
4	10108108004134	101081080	04134	ASSAULT, BATTERY	Sunday	11/21/2010	17:00	SOUTHERN	NONE	...	None	None	None	None	2	

5 rows x 35 columns

Problem solving and online analytical processing

What are the counts of the number of the crimes for different categories?

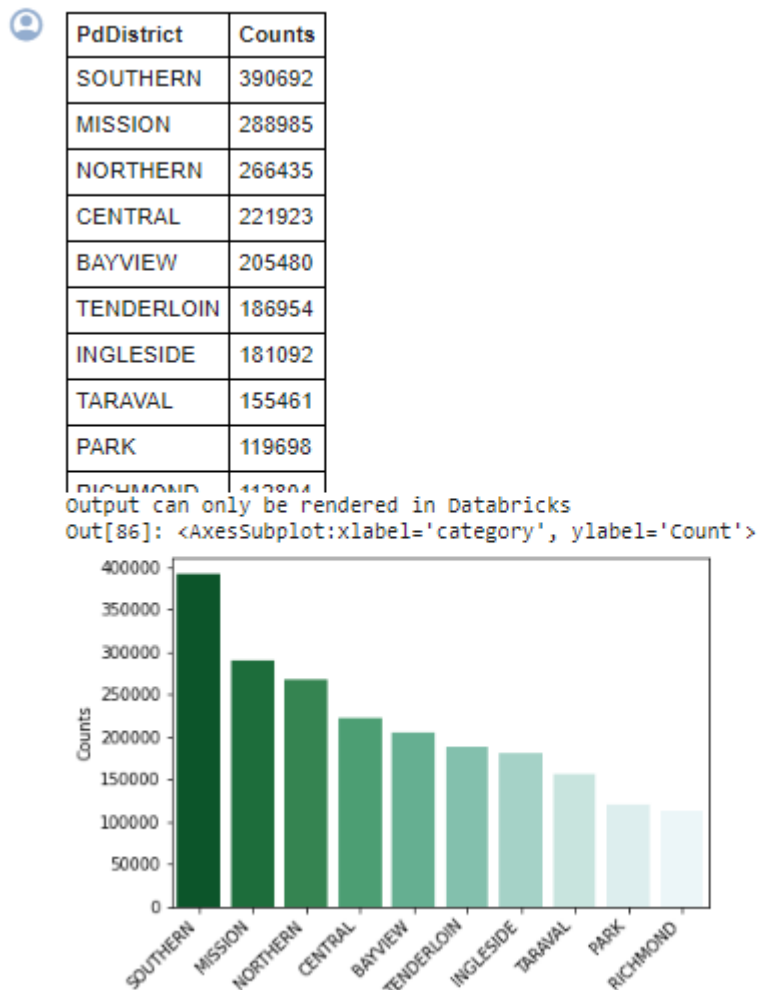
```
[ ] Out[80]: [Text(0, 0, 'LARCENY/THEFT'),
Text(1, 0, 'OTHER OFFENSES'),
Text(2, 0, 'NON-CRIMINAL'),
Text(3, 0, 'ASSAULT'),
Text(4, 0, 'VEHICLE THEFT'),
Text(5, 0, 'DRUG/NARCOTIC'),
Text(6, 0, 'VANDALISM'),
Text(7, 0, 'WARRANTS'),
Text(8, 0, 'BURGLARY'),
Text(9, 0, 'SUSPICIOUS OCC'),
Text(10, 0, 'ROBBERY'),
Text(11, 0, 'MISSING PERSON'),
Text(12, 0, 'FRAUD'),
Text(13, 0, 'FORGERY/COUNTERFEITING'),
Text(14, 0, 'SECONDARY CODES')]
```



According to the number of occurrence, they have of code analyse data frame for determining top 15 crime categories. The bar chart are used to display the results. The below is the interpretation and breakdown of the results. With the particular focus on top 15, the study aims to comprehend the distribution of crime categories within the data set (Revathi *et al.* 2023). It is clear from the interpretation of the graph that some Prime types are more common than others in the data set are.

"LARCENY/THEFT" seems to be the most commonly reported crime, indicating that it is a major issue in the area under analysis. For allocating resources and customising crime prevention practice, legislators and law enforcement must have an understanding of the distribution of the crime categories. In addition, law enforcement organisations may concentrate on putting specific measures into place to lower theft incidences if "LARCENY/THEFT" is particularly widespread.


Evaluate the number of the crimes for various district and visualise the findings




The code counts the number of the crimes, which have occurred in the various police districts by analysing a data frame using Spark SQL. The bar chart is used for displaying the results. The police district's total crime rate is summed up by the SQL query, which then sorts the results downward. The data frame displays the top police districts in terms of the total crime (Reddy *et al.* 2023). The data is represented in the following bar chart, where the y-axis shows the corresponding crime counts and x-axis has shown the police districts for improving visual appeal. Investigators and legislators can use this information to help them prioritising the distribution of resources and create

focused preventative plans for the regions with elevated crime that have been identified. The knowledge is communicated more readily according to the representation.


Analysis the number of crimes each Sunday at SF Downtown



Date	DayOfWeek	count
2003-01-05	Sunday	13
2003-01-12	Sunday	20
2003-01-19	Sunday	17
2003-01-26	Sunday	13
2003-02-02	Sunday	14
2003-02-09	Sunday	22
2003-02-16	Sunday	12
2003-02-23	Sunday	14
2003-03-02	Sunday	16



2003-03-02	Sunday	16
2003-03-09	Sunday	8
2003-03-16	Sunday	18
2003-03-23	Sunday	9
2003-03-30	Sunday	20
2003-04-06	Sunday	8
2003-04-13	Sunday	13
2003-04-20	Sunday	13
2003-04-27	Sunday	5
2003-05-04	Sunday	9



2003-08-31	Sunday	8
2003-09-07	Sunday	13
2003-09-14	Sunday	10
2003-09-21	Sunday	10
2003-09-28	Sunday	7
2003-10-05	Sunday	14
2003-10-12	Sunday	13
2003-10-19	Sunday	11
2003-10-26	Sunday	16
2003-11-02	Sunday	17

2004-07-25	Sunday	11
2004-08-01	Sunday	16
2004-08-08	Sunday	13
2004-08-15	Sunday	8
2004-08-22	Sunday	12
2004-08-29	Sunday	22
2004-09-05	Sunday	15
2004-09-12	Sunday	9
2004-09-19	Sunday	21
2004-09-26	Sunday	14

2005-05-29	Sunday	10
2005-06-05	Sunday	9
2005-06-12	Sunday	10
2005-06-19	Sunday	12
2005-06-26	Sunday	18
2005-07-03	Sunday	12
2005-07-10	Sunday	14
2005-07-17	Sunday	9
2005-07-24	Sunday	13
2005-07-31	Sunday	10


2018-03-11	Sunday	13
2018-03-18	Sunday	17
2018-03-25	Sunday	14
2018-04-01	Sunday	9
2018-04-08	Sunday	23
2018-04-15	Sunday	12
2018-04-22	Sunday	22
2018-04-29	Sunday	16
2018-05-06	Sunday	17
2018-05-13	Sunday	7

This code uses a spatial filter to find crimes that fall inside a given range that defines San Francisco's downtown. In addition, the SQL method selects and aggregates the crime counts according to the given temporal and spatial conditions using Spark SQL query (Guha Neogi *et al.* 2023). By utilising filtering conditions, data frame operations, DF method accomplishes the same goal. Both methods are offences that happen on Sunday through a specified range of San Francisco's downtown. In


addition, the generated Data frames are shown. When it comes to improving the accuracy and flexibility in the crime analysis, using UDFs for geographical filtering guarantees a personalised and effective way to analyse the crime data within particular geographical limits.

Analysis the number of crimes in every month of 2015 to 2018


```
#SQL way
crime_each_month=spark.sql("select sul
display(crime_each_month)
```



Years	Months	counts
2015	01	13181
2015	02	11882
2015	03	13463
2015	04	12526
2015	05	13318
2015	06	12853
2015	07	12949
2015	08	13317
2015	09	12476
2015	10	12607



2016	10	12913
2016	11	12254
2016	12	12629
2017	01	12687
2017	02	11780
2017	03	13250
2017	04	12452
2017	05	12758
2017	06	12186
2017	07	12717



2017	08	12428
2017	09	12204
2017	10	12970
2017	11	11940
2017	12	12115
2018	01	11667
2018	02	9565
2018	03	10354
2018	04	9954
2018	05	3519

The important insights into the temporal patterns can be gained from the examination of the crime statistics for every month in the years. According to the available data, it seems that the number of crimes varies from month to month within a year (Fernández Casaní *et al.* 2023). There is a discernible rise through the summer month of year 2015 whereas in 2018, there is a significant decrease in the crime rates in May. such tendencies may have big effects on the business, especially in the public safety, local business sector and tourists. The organisation can effectively allocate the resources by using this information during the periods of high crime. In addition, businesses can adjust their operation hours, security protocols, and marketing tactics by having a thorough grasp of crime trends. Understanding safer times of year can improve guest satisfaction and overall experience for the tourism sector. Overall, this analysis supports resource optimisation and strategic planning, making the community secure and safer.

Analysis crime numbers with respect to the hour on certain days

Date	Hour	count
2015-12-15	0	15
2015-12-15	1	6
2015-12-15	2	5
2015-12-15	3	4
2015-12-15	4	10
2015-12-15	5	3
2015-12-15	6	4
2015-12-15	7	8
2015-12-15	8	12

Date	Hour	counts
2017-12-15	14	11
2017-12-15	15	26
2017-12-15	16	30
2017-12-15	17	27
2017-12-15	18	28
2017-12-15	19	29
2017-12-15	20	17
2017-12-15	21	20
2017-12-15	22	36
2017-12-15	23	28

The DF Analysis looks into how many crimes occur with the given hour on the particular days. In this case for 10 December in 2015 to 2017. The generated data shows trends in the incidence of crimes on these specific dates during the course of the day. These kinds of information are essential for tourists who are organising their time in San Francisco. In order to have a safer experience, travellers may want to consider modifying their plans to avoid hours of high crime. This information can be held name for making decisions with scheduling visit to certain areas with them San Francisco improving the enjoyment and travel safety (Ramasami *et al.* 2023).

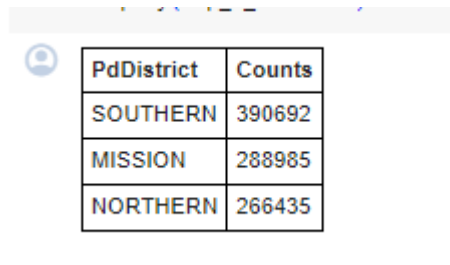
```
#SQL Way
crime_category=spark.sql("""
select Date,Hour,count(*) as counts
from sf_crime
where (Date='2015-12-15' or Date='2016-12-15' or Date='2017-12-15')
group by 1,2
order by 1,2""")
display(crime_category)
```

Date	Hour	counts
2015-12-15	0	15
2015-12-15	1	6
2015-12-15	2	5
2015-12-15	3	4
2015-12-15	4	10
2015-12-15	5	3
2015-12-15	6	4
2015-12-15	7	8
2015-12-15	8	12

The SQL study looks at how many crimes there are in relation to the hour on three particular days such December 15th in the year 2015, 2016 and year 2017. The data discussion theory covers

criminal activity and offers a picture of the crime patterns over these days. The statistics that are shown provide a clear picture of changes in crime over time, which might help those who are travelling through the city on certain dates make wise decisions.

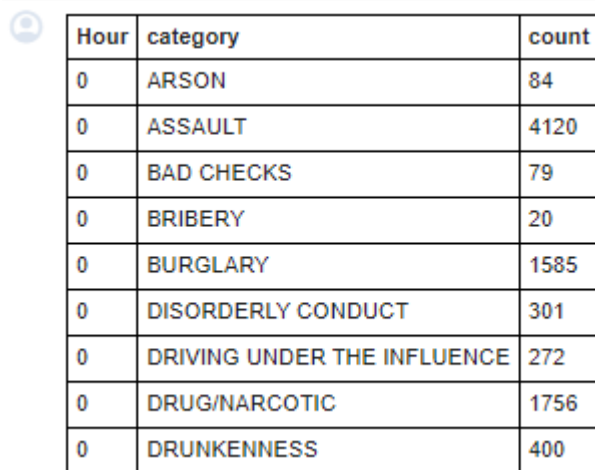
Finding out the crime event according to the time and category



A screenshot of a data table with a light blue header bar. To the left of the table is a small circular icon with a person silhouette. The table has two columns: 'PdDistrict' and 'Counts'. It lists three districts: SOUTHERN, MISSION, and NORTHERN, with their respective counts.

PdDistrict	Counts
SOUTHERN	390692
MISSION	288985
NORTHERN	266435

The top three districts with greatest crime counts are identified by the analysis utilising both SQL techniques. The crime events in such Districts are examined, trends that can inform the strategic deployment of police are reduced taken into account both time and category (Iatropoulou *et al.* 2023).



A screenshot of a data table with a light blue header bar. To the left of the table is a small circular icon with a person silhouette. The table has three columns: 'Hour', 'category', and 'count'. It lists ten categories of crime, all occurring at hour 0, with their respective counts.

Hour	category	count
0	ARSON	84
0	ASSAULT	4120
0	BAD CHECKS	79
0	BRIBERY	20
0	BURGLARY	1585
0	DISORDERLY CONDUCT	301
0	DRIVING UNDER THE INFLUENCE	272
0	DRUG/NARCOTIC	1756
0	DRUNKENNESS	400

According to the implementation of the designated top three districts, the data from analysis compiles crime counts and classifies them according to the type of crime. This detailed method aids in identifying the peak times for particular types of crimes in each district. For example, knowing that a certain category peaks at certain hours enables law enforcement to deploy resources efficiently during certain hours.

Find the resolution percentage for different categories of crime

```
[ ] # DF Way
category_resolution=df_opt1.filter(df_opt1['resolution']!='NONE').groupBy('category','resolution').count().orderBy('count',ascending=False)
display(category_resolution)
```

category	resolution	count
OTHER OFFENSES	ARREST, BOOKED	115427
DRUG/NARCOTIC	ARREST, BOOKED	97926
OTHER OFFENSES	ARREST, CITED	94817
WARRANTS	ARREST, BOOKED	93092
ASSAULT	ARREST, BOOKED	49246
NON-CRIMINAL	PSYCHOPATHIC CASE	27027
LARCENY/THEFT	ARREST, BOOKED	25136
MISSING PERSON	LOCATED	19615
WEAPON LAWS	ARREST, BOOKED	13334

```
[ ] # SQL Way
category_resolution=spark.sql("""
select category,resolution,count(*) as counts
from sf_crime
where resolution!='NONE'
group by 1,2
order by 3 desc
""")
display(category_resolution)
```

RECOVERED VEHICLE	NOT PROSECUTED	1
PROSTITUTION	NOT PROSECUTED	1
DRUNKENNESS	PROSECUTED BY OUTSIDE AGENCY	1
BAD CHECKS	ARREST, CITED	1
GAMBLING	DISTRICT ATTORNEY REFUSES TO PROSECUTE	1
LARCENY/THEFT	PROSECUTED FOR LESSER OFFENSE	1
DRUNKENNESS	PROSECUTED FOR LESSER OFFENSE	1
RECOVERED VEHICLE	EXCEPTIONAL CLEARANCE	1
SUICIDE	DISTRICT ATTORNEY REFUSES TO PROSECUTE	1
FORGERY/COUNTERFEITING	LOCATED	1
LOITERING	PROSECUTED BY OUTSIDE AGENCY	1

The analysis has shown resolution rates and categories using SQL methods and Data frame. Three categories are discussed with the greeted resolution rates, demonstrating successful law enforcement involvement through such cases. In addition, the realisation has important ramifications for how law enforcement tactics and resources are allocated. The general community safety and resolution rates are improved by increasing police presence in the high crime areas and putting focused the measurement of the crime prevention in place (Zemnickis, 2023). The results of the investigation has highlighted how crucial data driven decision making is to maximise the law enforcement operations and guaranteeing successful public safety outcomes.

Discussion/Findings

From the analysis, the research paper has explored different types of crime data using the implementation of the Apache spark SQL. The Research report has started to import the necessary packages and generate a spark data frame. Several steps involve online analytical processing and problem solving (Damus Ros, 2023).

- The number of crimes are counted for the different categories.
- Defining the crime count for various district and visualise the result
- Counting the crime numbers on Sunday at SF Downtown
- Analysis the number of crimes in every month of 2015 to 2018
- Analysis crime numbers with respect to the hour on certain days
- Finding out the crime event according to the time and category
- Find the resolution percentage for different categories of crime

Task 2: Implement machine learning algorithms for data analytics

Introduction of Problem

The research paper has discussed the use of machine learning classification algorithms to important data science difficulties is the essence of current problems. The goal is to use Spark Mallis for building classification models on the Titanic data set. Classification has played a key part in solving the real life problems and developing categorisation of the samples into established classes. In addition, the data set, which is best on the historical passenger information from the Titanic, offers an example of this situation in which predictive modelling can be quite helpful. The raw data choosing features preparing the structure data and putting the classification system into practice at the difficult parts of editing the survival outcomes. Taking advantage of the capabilities for Spark Mallis, this Topic addresses the significance of the categorization in the data science application and the decision making process by encapsulating the difficulty of Creative effective models from the diverse data sets.

Approaches

Two essential methods for solving the Titanic dataset classification problem with Spark Mallis are defined as the data pre-processing and feature engineering. Initially, preparing data for the machine-learning algorithm has record cleaning, resolving machine values and organising categorical values in the structure format. In addition, the future engineering place important role in the selection and transformation of the pertinent features that enhance the predictive accuracy. In order to represent

the information, this entails determining important attributes and producing a feature vector. By utilising the features of the Spark MLlib, such Methods come together to create article plan for developing classification model that can forecasts to the survival outcomes. Such procedures highlight how crucial careful features selection and data preparation are two maximize the machine learning model performance on the real world datasets (Singhal *et al.* 2023).

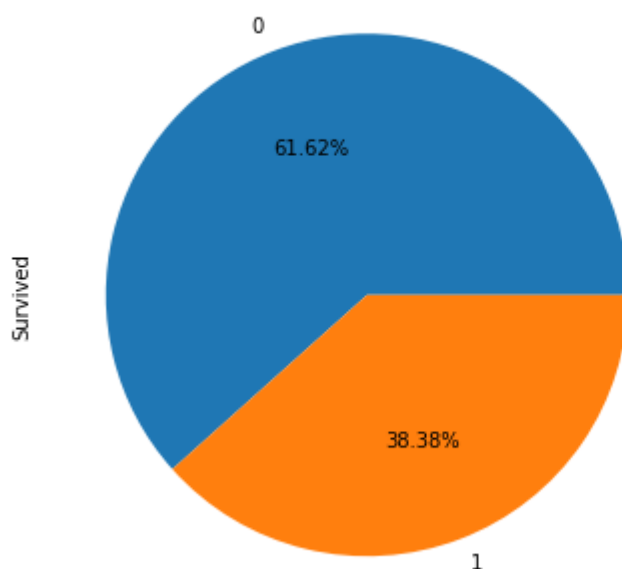
Results Analysis

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

The data has provided ID, survival status, name, gender, places, and number of children, parents, embarkation points, cabin, and fare and ticket information. The dataset is available for download. The vital data needed to investigate the traits and demography of titanic passengers is contained in this collection. Understanding the elements affecting the chances of surviving throughout and completely through, right the way through. In every part; everywhere. During an entire period, the whole time. Characteristics such as ticket class, gender and age may reveal trends, which have affected the survival rates. The dataset must be carefully pre-processed as it has contained the missing values for providing the model construction and reliable analysis (Eduru *et al.* 2023).

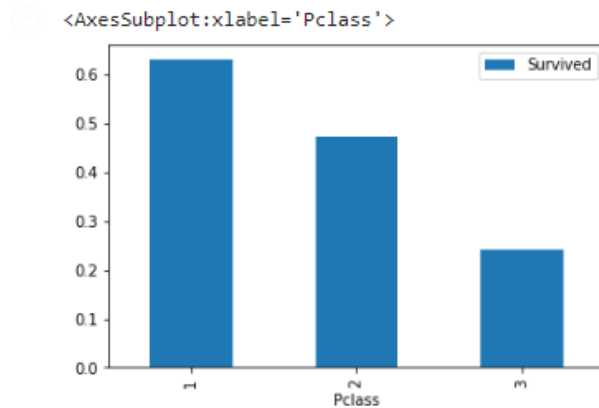


```
<AxesSubplot:ylabel='Survived'>
```



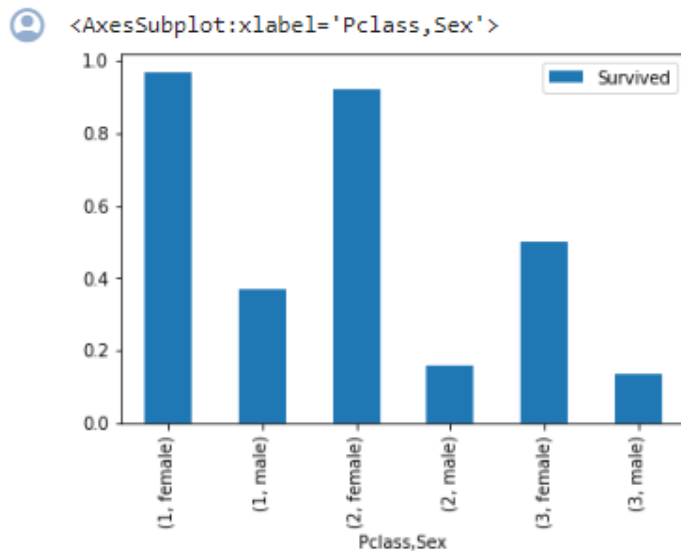
The pie chart Indicates 38.8% of the data side participants did not survive and 61.62% candidates survived. This implies that day survival rate changes significantly depending on Gender (Sioutas and Zaroliagis, 2023).

```
[ ] train_data[['Pclass', 'Survived']].groupby(['Pclass']).mean().plot.bar()
```



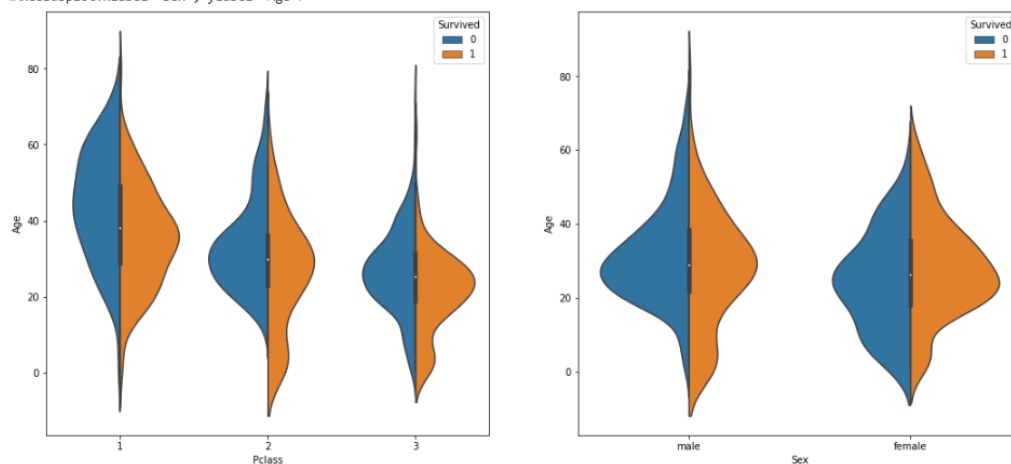
The above implementation has discussed the survival outcomes and passenger class which is shown accompanying the bar plot (Amruth *et al.* 2023). The main survival rate for individual passenger classes significantly shown in the above chart. From the above implementation, it has been stated that the passenger class has this client and this survival rate Has Fallen, which has suggested significant relationship between the chance of survival and higher class. The impact of the passenger class on the survival probability throughout the graphic, which is provided inside full information on the social economical dynamics of the Titanic data.

```
train_data[['Sex', 'Pclass', 'Survived']].groupby(['Pclass', 'Sex']).mean().plot.bar()
```

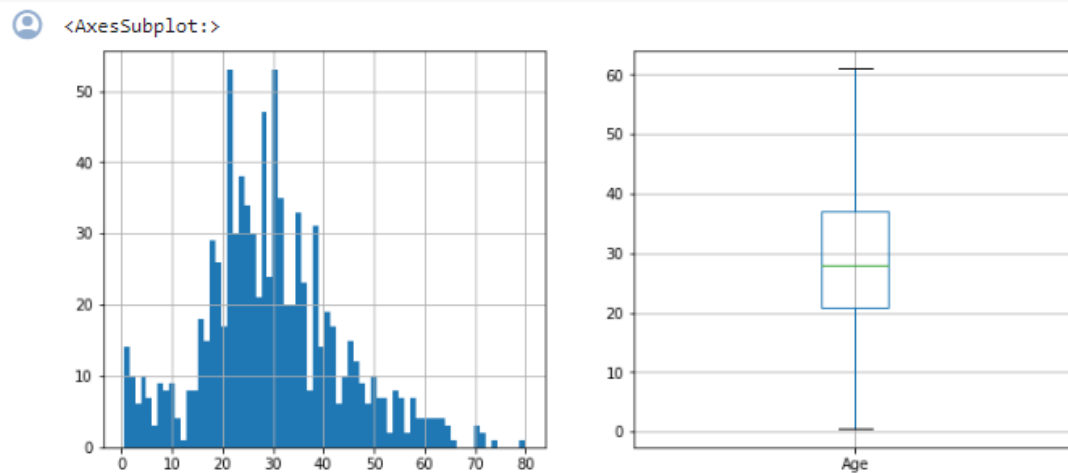


The court has used mat ply for creating bad blood for visualising the survival rate throughout the different passenger class and gender's. Those variables are combined for representing individual bar and the bar height has shown the main survival rate (Sarker, 2023). This graph clearly depicts the survival friends and provides information about how the persons are class and gender are affected survival rate. That through understanding of such characteristics are affected the probability of the Titanic survival which can be obtained by comparing the main survival rates across various demographic grouping using the plot interpretation.

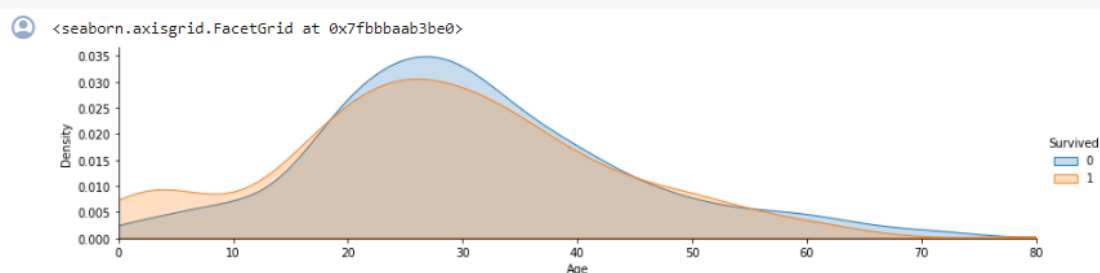
```
warnings.warn(
/Users/tamanna/opt/anaconda3/envs/env/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args:
warnings.warn(
<AxesSubplot:xlabel='Sex', ylabel='Age'>
```



This code has represented Gender and passenger class according to the link between survival status and age. The visualisation code has generated a comparison of the plots. The plots offer perspectives on such variables affecting life by illuminating the ways in which the age variable has influenced the survival outcomes through different gender and class groups (Zaki *et al.* 2024).



The age distribution has created using matplotlib visualisation function. The histogram has provided overview of the age frequency and the boxer has highlighted Central tendency for the age distribution and variability while removing outliers. Such visualisation has offered supplementary insights according to the age related features of the data set.



For investigating the relationship between the survival status and age, the visualisation court has discussed various colours corresponding to the different survival outcomes for individual component. That distribution of the age for the passengers who are survived are significantly used density plot. The visualise differentiation is improved by the shift parameter (Bharadiya, 2023). This is ensured that the graphic representation has discussed age related trends in both non-survivors and survivors. For insuring through perspective,

educational examination of the process of ageing and survival are discussed according to the data analysis.

✓ 6.2.1 Logistic regression

```
[ ] logreg = LogisticRegression()  
    logreg.fit(X_train, Y_train)  
    Y_pred = logreg.predict(X_test)  
    acc_log = round(logreg.score(X_train, Y_train) * 100, 2)  
    acc_log
```

 80.93

✓ 6.2.2 Support vector machines

```
[ ] svc = SVC()  
    svc.fit(X_train, Y_train)  
    Y_pred = svc.predict(X_test)  
    acc_svc = round(svc.score(X_train, Y_train) * 100, 2)  
    acc_svc
```


83.09

✓ 6.2.3 K-nearest neighbours (KNN)

```
[ ] knn = KNeighborsClassifier(n_neighbors = 5)  
    knn.fit(X_train, Y_train)  
    Y_pred = knn.predict(X_test)  
    acc_knn = round(knn.score(X_train, Y_train) * 100, 2)  
    acc_knn
```

84.79

✓ 6.2.4 Gaussian naive bayes

```
 gaussian = GaussianNB()  
    gaussian.fit(X_train, Y_train)  
    Y_pred = gaussian.predict(X_test)  
    acc_gaussian = round(gaussian.score(X_train, Y_train) * 100, 2)  
    acc_gaussian
```

 79.11

✓ 6.2.6 Linear SVC

```
[ ] linear_svc = LinearSVC()
    linear_svc.fit(X_train, Y_train)
    Y_pred = linear_svc.predict(X_test)
    acc_linear_svc = round(linear_svc.score(X_train, Y_train) * 100, 2)
    acc_linear_svc
```

80.25

✓ 6.2.7 Stochastic gradient descent

```
▶ sgd = SGDClassifier()
  sgd.fit(X_train, Y_train)
  Y_pred = sgd.predict(X_test)
  acc_sgd = round(sgd.score(X_train, Y_train) * 100, 2)
  acc_sgd
```

77.64

✓ 6.2.8 Decision tree

```
[ ] decision_tree = DecisionTreeClassifier()
    decision_tree.fit(X_train, Y_train)
    Y_pred = decision_tree.predict(X_test)
    acc_decision_tree = round(decision_tree.score(X_train, Y_train) * 100, 2)
    acc_decision_tree
```

85.7

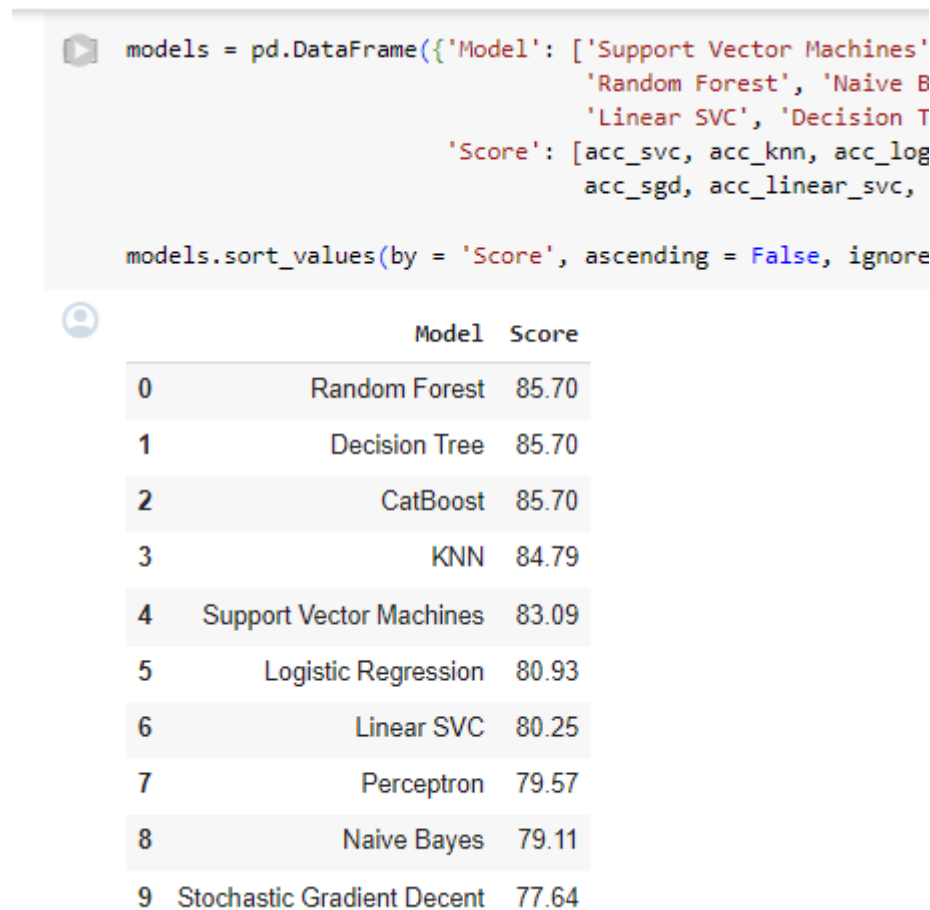
✓ 6.2.9 Random forest

```
[ ] random_forest = RandomForestClassifier(n_estimators = 100)
    random_forest.fit(X_train, Y_train)
    Y_pred = random_forest.predict(X_test)
    acc_random_forest = round(random_forest.score(X_train, Y_train) * 100, 2)
    acc_random_forest
```

85.7

The code has created a data frame for summarising the machine learning model performance. The names of different algorithms listed in the above section to implement accuracy scores. The data frame can be sorted by the evaluating accuracy value to facilitate

comparison of the model performance. With the help of this tabular format, which offers a clear summary to facilitate well-informed decision-making when choosing the best method for the provided data set and issue, which involves it is possible to determine which models are most useful for the current work.



```
models = pd.DataFrame({'Model': ['Support Vector Machines', 'Random Forest', 'Naive B', 'Linear SVC', 'Decision T', 'Score': [acc_svc, acc_knn, acc_log, acc_sgd, acc_linear_svc, models.sort_values(by = 'Score', ascending = False, ignore
```

	Model	Score
0	Random Forest	85.70
1	Decision Tree	85.70
2	CatBoost	85.70
3	KNN	84.79
4	Support Vector Machines	83.09
5	Logistic Regression	80.93
6	Linear SVC	80.25
7	Perceptron	79.57
8	Naive Bayes	79.11
9	Stochastic Gradient Decent	77.64

The data has presented the value of different machine learning models on the particular section. With accuracy value of 85.70%, decision tree, random forest along with Cat Boost has been defined as the best model. The summary has helped for choosing the best algorithm with the decision tree, Cat boost and Random Forest for the given dataset. Such results are directed to additional research and model optimisation for best performance to address underlying issues and dataset properties.

Discussion and conclusions

The Titanic data has provided important insight into the social economical dynamics during the event. The relationship between survival outcomes and passenger class is the subject of the research analysis. The breakdown of the non-survival and survival instance within

individual class is revealed by the examine the survival counts which is grouped by the Classes. The impact of the social economical position on the survival probabilities is significantly visualised by the bar plot showing the main survival rate by the passenger class (Rani *et al.* 2023). The pattern, which has been shown, shows the clear relationship between the survival rate and higher passenger class. The first passenger class have highest means survival rate, which is followed by the third class passengers and second-class passengers. It is consist and with the historical narratives of The Titanic accident, wherein passengers of better class were frequently granted precedence in gaining access to lifesaving vessels and aid operations. The social economical differences of the data set are a reflection of the different circumstances that Titanic passengers encountered. The visually bar plot has highlighted the advantage of the first class passengers which have in terms of the survival chances. This analysis provides as a stinging reminder of the disparities that persist throughout catastrophes, in addition to confirming historical narratives. Moreover, survival outcomes and Class are export in a way that goes beyond the statistical patterns. It raises questions about more general societal issues by highlighting the moral dilemmas associated with resource allocation and crisis response possession of life-saving interventions (Zaki *et al.* 2024). These kinds of analyses are crucial for comprehending the historical background and add to the current conversations about responses to emergencies and social fairness. The graphical display of average survival rates among various passenger classes is an effective means of conveying historical facts. Beyond only looking at statistical trends, this investigation raises questions about moral dilemmas and social injustices. It emphasises how crucial it is to use data analysis to not only unearth historical secrets but also to add to the conversation about social justice and humanitarian concerns in emergencies.

References and Bibliography

Amruth, A., Ramanan, R., Paul, R., Vishal, S. and Saravanan, S., 2023, June. Big Data Application in Cancer Classification by Analysis of RNA-seq Gene Expression. In *2023 3rd International Conference on Intelligent Technologies (CONIT)* (pp. 1-6). IEEE.

Bharadiya, J.P., 2023. A comparative study of business intelligence and artificial intelligence with big data analytics. *American Journal of Artificial Intelligence*, 7(1), p.24.

Bharadiya, J.P., 2023. Machine learning and AI in business intelligence: Trends and opportunities. *International Journal of Computer (IJC)*, 48(1), pp.123-134.

Damus Ros, N., 2023. A Business Intelligence Solution, based on a Big Data Architecture, for processing and analyzing the World Bank data.

Eduro, H.V., Vivek, Y., Ravi, V. and Shankar, O.S., 2023. Parallel and streaming wavelet neural networks for classification and regression under apache spark. *Cluster Computing*, pp.1-19.

Fernández Casaní, Á., García Montoro, C., González de la Hoz, S., Salt, J., Sánchez, J. and Villaplana Pérez, M., 2023. Big Data Analytics for the ATLAS EventIndex Project with Apache Spark. *Computational and Mathematical Methods*, 2023.

Fernando, S., Mydlarz, V.S., Katanani, A. and Virdee, B., 2023, June. Cloud Spark Cluster to analyse English prescription big data for NHS intelligence. In *International Conference on Data Analytics & Management* (pp. 361-375). Singapore: Springer Nature Singapore.

Guha Neogi, A., Eltaher, A. and Sargsyan, A., 2023. NGS Data Analysis with Apache Spark. In *Computational Life Sciences: Data Engineering and Data Mining for Life Sciences* (pp. 441-467). Cham: Springer International Publishing.

Iatropoulou, S., Anastasiou, T., Karagiorgou, S., Petrou, P., Alexandrou, D. and Bouras, T., 2023, April. Privacy-preserving Data Federation for Trainable, Queryable and Actionable Data. In *2023 IEEE 39th International Conference on Data Engineering Workshops (ICDEW)* (pp. 44-48). IEEE.

Ramasami, M.V., Thangaraj, R., Kumar, S.M. and Eswaran, S., 2023, July. Exploratory Data Analysis of Walmart Outlets Sales using Data Analytics Techniques. In *2023 International Conference on Digital Applications, Transformation & Economy (ICDATE)* (pp. 1-4). IEEE.

Rani, P., Lamba, R., Sachdeva, R.K., Kumar, R. and Bathla, P., 2023. Big Data Analytics: Integrating Machine Learning with Big Data Using Hadoop and Mahout. *Intelligent Systems and Smart Infrastructure: Proceedings of ICISSI 2022*, p.366.

Reddy, N. and Peneti, S., 2023, May. Advanced lab analysis system using apache spark. In *AIP Conference Proceedings* (Vol. 2492, No. 1). AIP Publishing.

Revathi, R., Alzeyadi, A.K., Hasan, H.M., Lafta, A.A., Balachander, B. and Shankar, B.B., 2023, September. A real-time big data architecture for covid dataset analysis with query on spark. In *AIP Conference Proceedings* (Vol. 2845, No. 1). AIP Publishing.

Sarker, I.H., 2023. Machine learning for intelligent data analysis and automation in cybersecurity: current and future prospects. *Annals of Data Science*, 10(6), pp.1473-1498.

Singhal, P. and Yadav, R.K., 2023. Electrocardiogram Feature Based Heart Arrhythmia Detection Using Machine Learning and Apache Spark.

Sioutas, S. and Zaroliagis, C., SQL Query Optimization in Distributed NoSQL Databases for Cloud-Based Applications.

Zaki, A.M., Khodadadi, N., Lim, W.H. and Towfek, S.K., 2024. Predictive Analytics and Machine Learning in Direct Marketing for Anticipating Bank Term Deposit Subscriptions. *American Journal of Business and Operations Research*, 11(1), pp.79-88.

Zemnickis, J., 2023. Data Warehouse Data Model Improvements from Customer Feedback. *Baltic Journal of Modern Computing*, 11(3).

Appendix (if necessary)

Appendices 1

1.1 Import Package

```
from csv import reader
from pyspark.sql import Row
from pyspark.sql import SparkSession
from pyspark.sql.types import *
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
import warnings

import os
os.environ["PYSPARK_PYTHON"] = "python3"
```

```
# Import data from SF government official website
import urllib.request
urllib.request.urlretrieve("https://data.sfgov.org/api/views/tmnf-yvry/rows.csv?accessType=DOWNLOAD", "/tmp/myxxxx.csv")
dbutils.fs.mv("file:/tmp/myxxxx.csv", "dbfs:/laioffer/spark_hw1/data/sf_03_18.csv")
display(dbutils.fs.ls("dbfs:/laioffer/spark_hw1/data/"))
```

path	name	size	modificationTime
dbfs:/laioffer/spark_hw1/data/sf_03_18.csv	sf_03_18.csv	550945238	1675130352000

Appendices 2

1.2 Generate Spark Object

```
[ ] from pyspark.sql import SparkSession

# Generate Spark Object from CSV
spark = SparkSession \
    .builder \
    .appName("crime analysis") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()

df_opt1 = spark.read.format("csv").option("header", "true").load(data_path)
df_opt1.createOrReplaceTempView("sf_crime")

df_opt1_pd = df_opt1.limit(5)
df_opt1_pd = df_opt1_pd.toPandas()
df_opt1_pd.head(10)
# display(df_opt1)
```


1.3 Basic Data Cleaning

```
[ ] from pyspark.sql.functions import *
    ##change datatype from string to Date/Month/Year

    df_opt1 = df_opt1.withColumn("Date", to_date(col("Date"), "MM/dd/yyyy"))
    df_opt1 = df_opt1.withColumn("Month", month(df_opt1['Date']))
    df_opt1 = df_opt1.withColumn("Year", year(df_opt1['Date']))

    df_opt1.createOrReplaceTempView("sf_crime")

    df_opt1_pd = df_opt1.limit(5)
    df_opt1_pd = df_opt1_pd.toPandas()
    df_opt1_pd.head(10)
    #display(df_opt1)
```

Appendices 3

Part 2: Online Analytical Processing and Problems Solving

Q1 What are the counts of numbers of crimes for different category?

```
q1_result = df_opt1.groupBy('category').count().orderBy('count', ascending=False)

q1_result_pd = q1_result.limit(15).toPandas()

q1_result_chart = sns.barplot(x='category', y='count', palette='BuGn_r', data=q1_result_pd)
q1_result_chart.set_xticklabels(q1_result_chart.get_xticklabels(), rotation=45, horizontalalignment='right')
#display(q1_result_pd)
```

```
Out[80]: [Text(0, 0, 'LARCENY/THEFT'),
Text(1, 0, 'OTHER OFFENSES'),
Text(2, 0, 'NON-CRIMINAL'),
Text(3, 0, 'ASSAULT'),
Text(4, 0, 'VEHICLE THEFT'),
Text(5, 0, 'DRUG/NARCOTIC'),
Text(6, 0, 'VANDALISM'),
Text(7, 0, 'WARRANTS'),
Text(8, 0, 'BURGLARY'),
Text(9, 0, 'SUSPICIOUS OCC'),
Text(10, 0, 'ROBBERY'),
Text(11, 0, 'MISSING PERSON'),
Text(12, 0, 'FRAUD'),
Text(13, 0, 'FORGERY/COUNTERFEITING'),
Text(14, 0, 'SECONDARY CODES')]
-----
```

Appendices 4

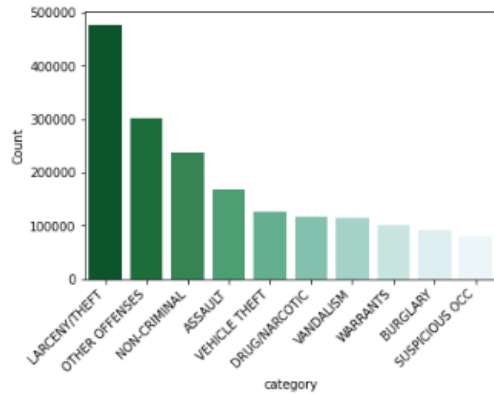
```

crimeCategory = spark.sql("SELECT category, COUNT(*) AS Count FROM sf_crime GROUP BY category ORDER BY Count DESC")

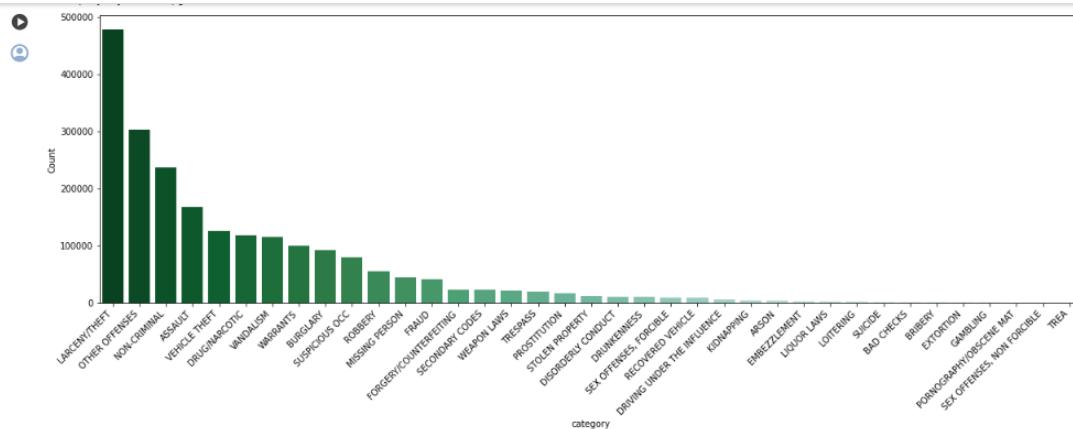
crimeCategory_pd = crimeCategory.limit(10).toPandas()
crimeCategory_chart = sns.barplot(x='category', y='Count', palette='BuGn_r', data=crimeCategory_pd)
crimeCategory_chart.set_xticklabels(crimeCategory_chart.get_xticklabels(), rotation=45, horizontalalignment='right')
crimeCategory_chart

```

Out[82]: <AxesSubplot:xlabel='category', ylabel='Count'>



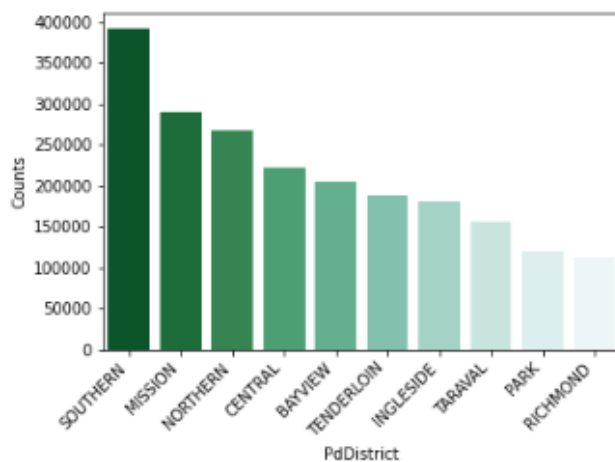
Appendices 5



NORTHERN	266435
CENTRAL	221923
BAYVIEW	205480
TENDERLOIN	186954
INGLESIDE	181092
TARAVAL	155461
PARK	119698
RICHMOND	112804
NA	1

Output can only be rendered in Databricks

Out[86]: <AxesSubplot:xlabel='category', ylabel='Count'>

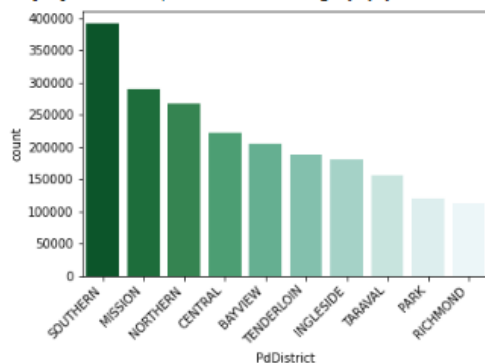


Appendices 6

```
# DF way
crimedistrict=df_opt1.groupBy('PdDistrict').count().orderBy('Count',ascending=False)

crimedistrict_pd = crimedistrict.limit(10).toPandas()
crimedistrict_chart = sns.barplot(x='PdDistrict', y = 'count', palette = 'BuGn_r', data = crimedistrict_pd)
crimedistrict_chart.set_xticklabels(crimedistrict_chart.get_xticklabels(), rotation=45, horizontalalignment='right')
crimecategory_chart
```

Out[91]: <AxesSubplot:xlabel='category', ylabel='Count'>



```
# DF Way
crime_sunday_SFdown=df_opt1.filter((df_opt1.X > -122.4313) & (df_opt1.X < -122.4313))
display(crime_sunday_SFdown)
```

2018-01-14	Sunday	10
2018-01-21	Sunday	19
2018-01-28	Sunday	13
2018-02-04	Sunday	12
2018-02-11	Sunday	15
2018-02-18	Sunday	8
2018-02-25	Sunday	15
2018-03-04	Sunday	22
2018-03-11	Sunday	13
2018-03-18	Sunday	17
2018-03-25	Sunday	14

Appendices 7

```
%sql select substring(Date,1,4) as Years,substring(Date,6,2) as Months,count(*) as counts from sf_crime where (Date>='2015-01-01' and Date<='2018-12-31') group by 1,2 order by 1,2
```

2017	08	12428
2017	09	12204
2017	10	12970
2017	11	11940
2017	12	12115
2018	01	11667
2018	02	9565
2018	03	10354
2018	04	9954
2018	05	3519

```
[ ] #DF way
crime_category=df_opt1[df_opt1.Year.isin([2015,2016,2017,2018])].groupBy('Year','Category').count().orderBy('Year','Category',ascending=True)
display(crime_category)
```

2015	VEHICLE THEFT	7935
2015	WARRANTS	6724

```
[ ] from pyspark.sql.functions import to_timestamp
```

```
# add new columns to convert Time to hour format
df_opt1 = df_opt1.withColumn('Time', to_timestamp(df_opt1['Time'], 'HH:mm'))

# extract hour from incident time
df_opt1 = df_opt1.withColumn('Hour', hour(df_opt1['Time']))
#display(df_opt1)
df_opt1.createOrReplaceTempView("sf_crime")
```

```
[ ] #DF way
```

```
crime_category=df_opt1[df_opt1.Date.isin(['2015-12-15', '2016-12-15', '2017-12-15'])].groupBy('Date', 'Hour')
display(crime_category)
```

Date	Hour	Count
2017-12-15	14	11
2017-12-15	15	26
2017-12-15	16	30
2017-12-15	17	27
2017-12-15	18	28
2017-12-15	19	29
2017-12-15	20	17
2017-12-15	21	20
2017-12-15	22	36

Appendices 8



#SQL Way

```
top_3_district=spark.sql("select PdDistrict, count(*) as Counts from sf_crime group by PdDistrict order by Counts Desc limit 3")
display(top_3_district)
```



PdDistrict	Counts
SOUTHERN	390692
MISSION	288985
NORTHERN	266435

```
[ ] #DF Way
```

```
category_time_top3=df_opt1[df_opt1.PdDistrict.isin(['MISSION', 'NORTHERN', 'SOUTHERN'])].groupBy('Hour', 'category').count().orderBy('Hour', 'category', ascending=True)
display(category_time_top3)
```

Hour	category	count
0	ARSON	84
0	ASSAULT	4120
0	BAD CHECKS	79
0	BRIBERY	20
0	BURGLARY	1585
0	DISORDERLY CONDUCT	301
0	DRIVING UNDER THE INFLUENCE	272
0	DRUG/NARCOTIC	1756
0	DRUNKENNESS	400

adjust the policy.

```
# DF Way
category_resolution=df_opt1.filter(df_opt1['resolution']!='NONE').groupBy('category','resolution').count().orderBy('count',ascending=False)
display(category_resolution)
```

WEAPON LAWS	ARREST, BOOKED	13334
BURGLARY	ARREST, BOOKED	12765
LARCENY/THEFT	ARREST, CITED	10463
NON-CRIMINAL	ARREST, BOOKED	9918
STOLEN PROPERTY	ARREST, BOOKED	9693
PROSTITUTION	ARREST, CITED	9667
ROBBERY	ARREST, BOOKED	9598
DRUG/NARCOTIC	ARREST, CITED	9046
VANDALISM	ARREST, BOOKED	8734
SECONDARY CODES	ARREST, BOOKED	7811

```
[ ] # SQL Way
category_resolution=spark.sql("""
select category,resolution,count(*) as counts
from sf_crime
where resolution!='NONE'
group by 1,2
order by 3 desc
""")
```

Appendices 9

1- Import Data

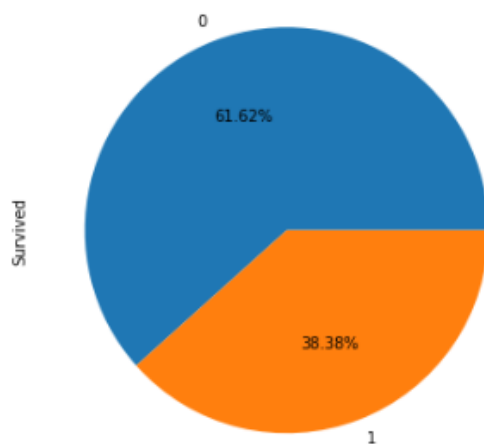
Load the training data first, then check the data format

```
train_data = pd.read_csv('data/train.csv')
data_test = pd.read_csv('data/test.csv')
train_data.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
# First look at the overall survival ratio
fig = plt.figure(figsize=(6,6))
train_data['Survived'].value_counts().plot.pie(autopct = '%1.2f%%')
```

<AxesSubplot:ylabel='Survived'>



Appendices 10

```
# Next, look at the complementary data  
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 891 entries, 0 to 890  
Data columns (total 12 columns):  
#   Column          Non-Null Count  Dtype  
---  ---  
0   PassengerId      891 non-null    int64  
1   Survived         891 non-null    int64  
2   Pclass          891 non-null    int64  
3   Name             891 non-null    object  
4   Sex              891 non-null    object  
5   Age              891 non-null    float64  
6   SibSp            891 non-null    int64  
7   Parch            891 non-null    int64  
8   Ticket           891 non-null    object  
9   Fare             891 non-null    float64  
10  Cabin            891 non-null    object  
11  Embarked         891 non-null    object  
dtypes: float64(2), int64(5), object(5)  
memory usage: 83.7+ KB
```

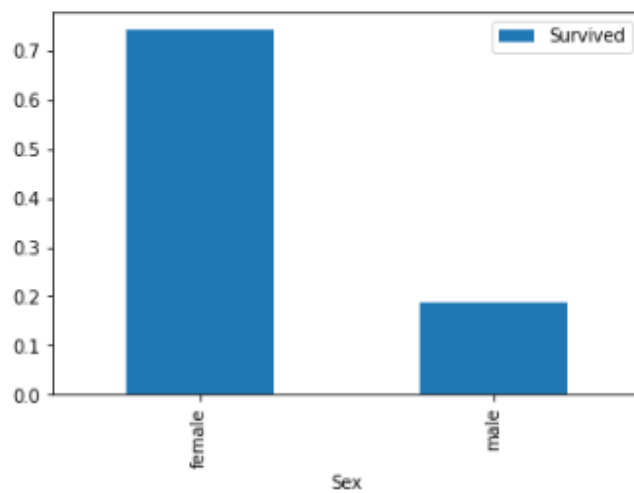
Appendices 11

```
[ ] train_data.groupby(['Sex', 'Survived'])['Survived'].count()
```

```
Sex    Survived
female 0         81
       1        233
male   0        468
       1        109
Name: Survived, dtype: int64
```

```
▶ survived_by_sex = train_data[['Sex', 'Survived']].groupby('Sex').mean()
type(survived_by_sex)
survived_by_sex.plot.bar()
```

```
ⓘ <AxesSubplot:xlabel='Sex'>
```



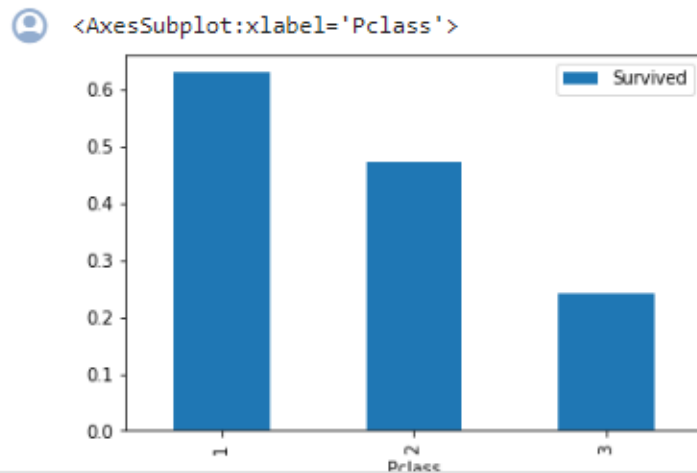
Appendices 12


```
[ ] train_data.groupby(['Pclass', 'Survived'])['Pclass'].count()
```

Pclass	Survived	
1	0	80
	1	136
2	0	97
	1	87
3	0	372
	1	119

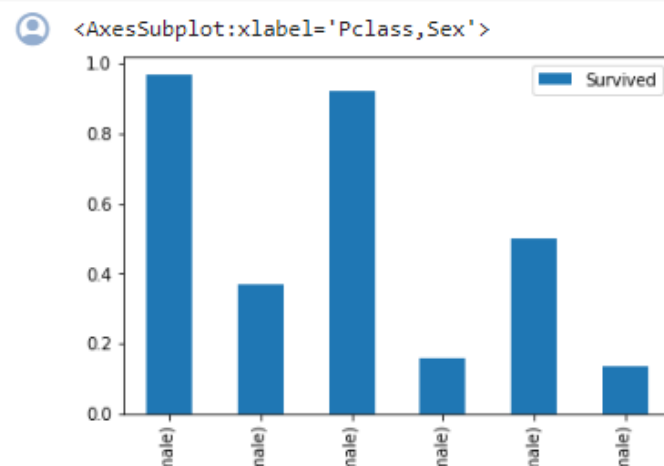
Name: Pclass, dtype: int64

```
▶ train_data[['Pclass', 'Survived']].groupby(['Pclass']).mean().plot.bar()
```

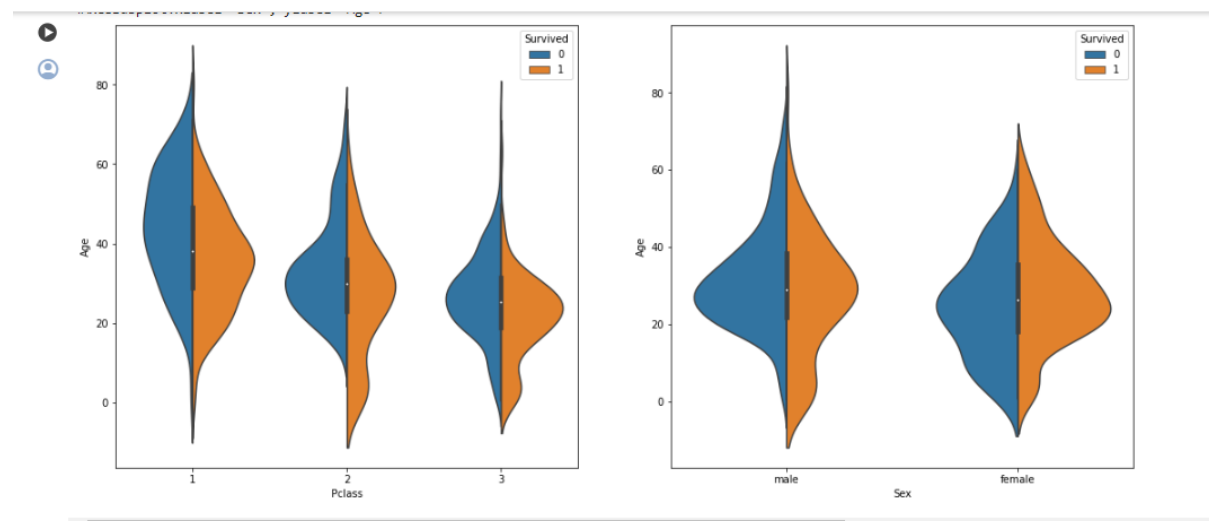


```
[ ] female 1      0      3
          1      91
          2      0      6
          1      70
          3      0      72
          1      72
male      1      0      77
          1      45
          2      0      91
          1      17
          3      0     300
          1      47
Name: Survived, dtype: int64
```

```
train_data[['Sex', 'Pclass', 'Survived']].groupby(['Pclass', 'Sex']).mean().plot.bar()
```

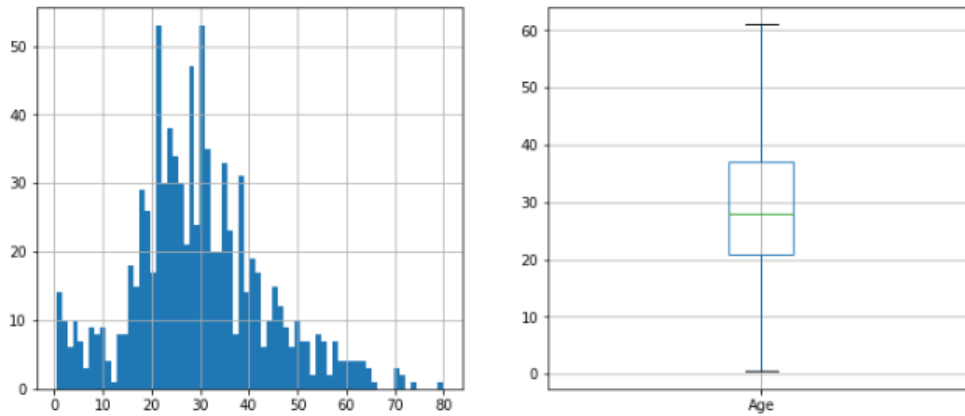


Appendices 13



```
[ ] plt.subplot(122)
    train_data.boxplot(column='Age', showfliers=False)
```

<AxesSubplot:>



Appendices 14

```
[ ] facet = sns.FacetGrid(train_data, hue='Survived', aspect=4)
    facet.map(sns.kdeplot, 'Age', shade=True)
    facet.set(xlim=(0, train_data['Age'].max()))
    facet.add_legend()
```

<seaborn.axisgrid.FacetGrid at 0x7fbbbaab3be0>

