BIG DATA (BD) COURSEWORK REPORT

Do not write your name on your work unless your lecturer has explicitly told you to do so.

Student ID number	Title of degree studying	**Level/Year**
2200918	Bachelor of Science (Honours) Data Science and Analytics	1

Short unit name:	M32364 – Big Data			Due date: 29 May 2023		Deadline: 29 May
Full unit name:	Big Data					
Unit lecturer name:				Group: *(if applicable)*		
Additional items e.g. CD/disk/USB:	Yes		No	✓	**Details:**	

All additional items should be clearly labelled with ID number and unit name and securely attached to your work.

Candidates are reminded that the following are defined as Assessment Offences and will be dealt with in accordance with the University's Code of Student Discipline:

a) Any attempt to complete an assessment by means considered to be unfair;

b) Plagiarism, which the University defines as the incorporation by a student in work for assessment of material which is not their own, in the sense that all or a substantial part of the work has been copied without any adequate attempt at attribution or has been incorporated as if it were the student's own work when in fact it is wholly or substantially the work of another person or persons.

Please note: Group **coursework** will be filed under the first Student ID number at the top of the list. Ensure you know all **group member's ID numbers.**

NB: **Coursework not collected** will be disposed of six months **after** the hand-in date.

FOR OFFICIAL USE ONLY

Date received/Office stamp	Provisional mark % / Comments
Administration Office	Academic Staff Member

1. Introduction – Titanic Dataset

The Titanic dataset contains information on passengers aboard the Titanic, including their demographics, cabin class, and survival status.

The problem statement for the Titanic dataset is to predict whether a passenger survived or not based on their demographic and other features. This is a binary classification problem. It provides an opportunity to explore various machine learning techniques for dealing with class imbalance and feature engineering. Additionally, the dataset has a historical significance and can be used to gain insights into the factors that influenced survival on the Titanic.

1.1 Details of Approach

- Supervised Learning

Since the target of this project is to predict whether the passenger will survive or not which is clearly a binary classification problem (target variable survived only contain 0 or 1). Hence for this project, 3 models and their respective optimised models will be applied to make predictions and then follow by the comparison of their classification performance to determine the ideal model. These 3 classification models are:

- Logistic Regression
- SVC – RBF (kernel with RBF)
- Random Forest Classifier (RFC)

I will be also using Convolutional Neural Network (CNN) model for experimenting to see whether if the result will be better than the classification models.

- Unsupervised Learning

The clustering model are then used to uncover segment attributes that is relating to survivors. The two clustering models are:

- Mean shift Clustering

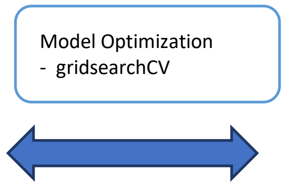
While Mean shift can work well for small datasets, it may have limitations when the number of features is high, or the dataset contains many redundant or irrelevant dimensions. However, for datasets with a low number of features, Mean shift can effectively identify clusters based on density gradients and can handle irregularly shaped clusters.

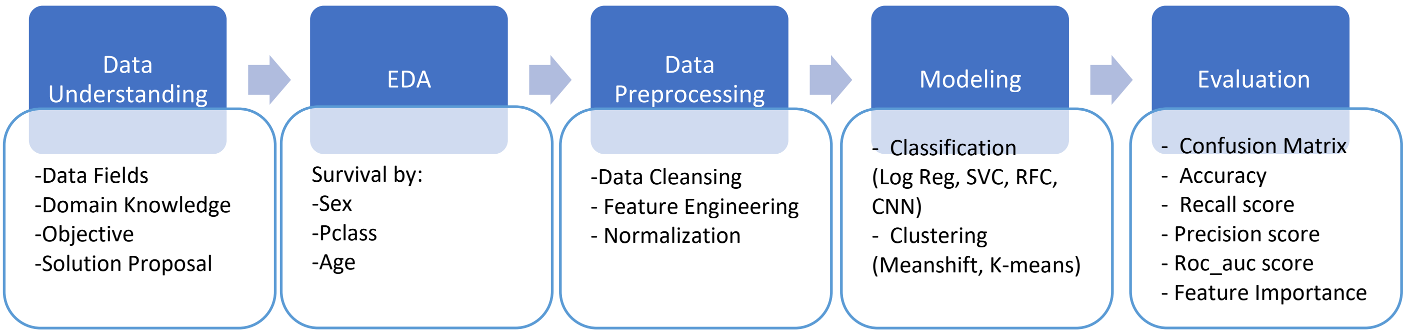
- K-means Clustering

K-means is a simple and computationally efficient algorithm that can be effective for small datasets. It assigns each data point to the nearest cluster centroid, making it suitable for small datasets with well-separated clusters. However, it is important to choose the appropriate number of clusters for effective results.

1.2 Algorithms

The overall solution approach could be summarized as below flowchart:





1.3 Experimental Results and Analysis

1.3.1 Experimental Setup

1. Data Understanding

```
#train set first 5 rows
data_train.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

There are 891 rows and 12 columns in Titanic dataset. Passenger demographics can be represented by Age, Sex, Pclass. There are missing values in Age, Cabin, and Embarked features. 2 float, 5 int, 5 string data types

Titantic Features Table

Features	Description
Survived	Survived (1) or died (0)
Pclass	Passenger's class
Name	Passenger's name
Sex	Passenger's sex
Age	Passenger's age
SibSp	Number of siblings/spouses aboard
Parch	Number of parents/children aboard
Ticket	Ticket number
Fare	Fare
Cabin	Cabin
Embarked	Port of embarkation (C-cherbourg, Q-Queenstown, S=Southampton)

```
data_train.info()
```

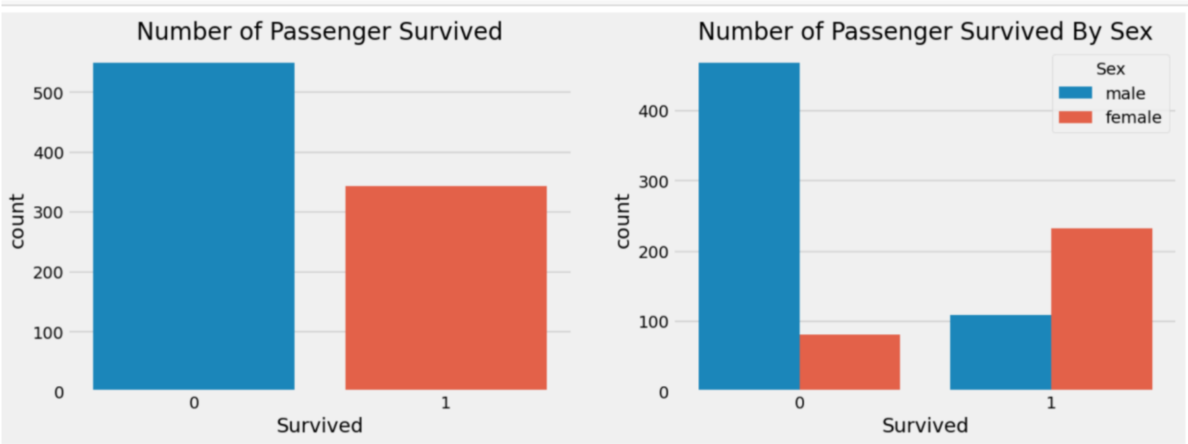
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 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        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

2. EDA

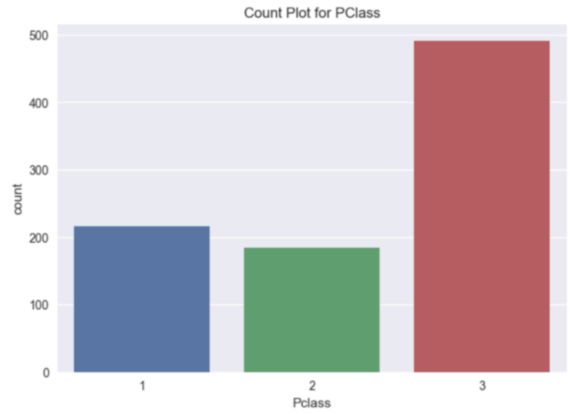
```
data_train.describe()
```

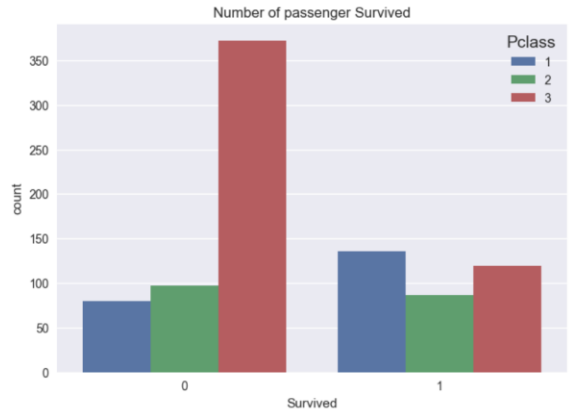
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Passengers had only average of 38% survival rate. An average fare of $32 and average age of 29.

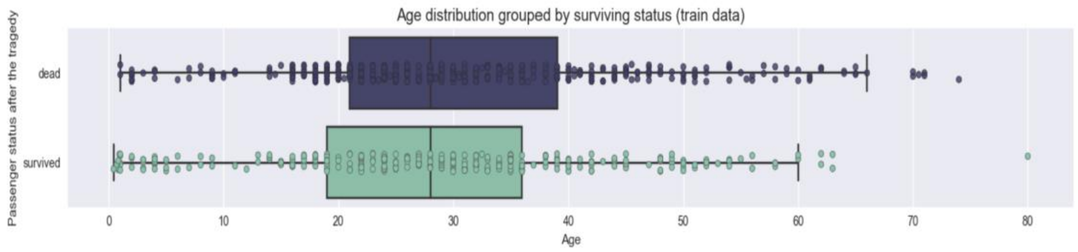


Plot above shown that passengers had a higher non-survival rate and Female are more likely to survive than Male.





Plot above shown majority of the passenger are from pclass 3 and pclass 3 had the highest death count. Pclass 1 had the highest survival count.



Box plot above shown there some outliers for survival (dead or survived)

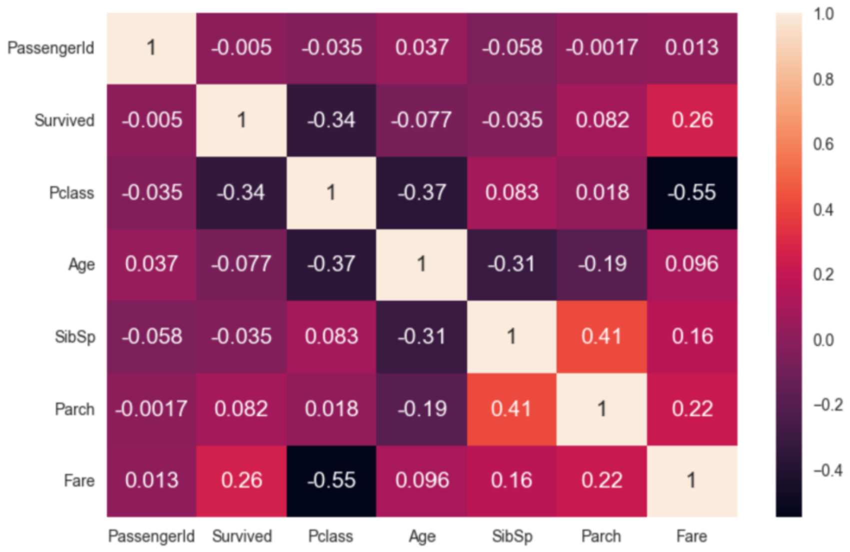
3. Data Pre-processing

- Data Cleansing

```
# Check any null vaues for train data
data_train.isna().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

177 missing values from Age, 687 missing values from cabin and 2 missing values from embarked.



- Pclass and age, as they had max relation in the entire set, filled up missing age values with median age calculated per class

```
# Age
data_train.loc[data_train.Age.isnull(), 'Age'] = data_train.groupby("Pclass").Age.transform('median')
data_test.loc[data_test.Age.isnull(), 'Age'] = data_test.groupby("Pclass").Age.transform('median')
```

- Age feature change from float to int data type

```
data_train['Age'] = data_train['Age'].astype(int)
data_test['Age'] = data_test['Age'].astype(int)
```

- Fare feature round up to 2 decimal places.

```
data_train['Fare'] = data_train['Fare'].round(2)
data_test['Fare'] = data_test['Fare'].round(2)
```

- Filling missing values of fares feature with median fares per class

```
# Fare
data_train['Fare']  = data_train.groupby("Pclass")['Fare'].transform(lambda x: x.fillna(x.median()))
data_test['Fare']  = data_test.groupby("Pclass")['Fare'].transform(lambda x: x.fillna(x.median()))
```

- Filling missing values of embarked feature with mode

```
data_train['Embarked'] = data_train['Embarked'].fillna(mode(data_train['Embarked']))
data_test['Embarked'] = data_test['Embarked'].fillna(mode(data_test['Embarked']))
```

- Filling missing values of cabin feature with unknown

```
data_train['Cabin'] = data_train['Cabin'].fillna('Unknown')
data_test['Cabin'] = data_test['Cabin'].fillna('Unknown')
```

After data cleaning, no more missing values found.

```
data_train.isna().sum()
```

```
PassengerId    0
Survived       0
Pclass         0
Name           0
Sex            0
Age            0
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin          0
Embarked       0
dtype: int64
```

```
data_train.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    int32
 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(1), int32(1), int64(5), object(5)
memory usage: 80.2+ KB
```

- Feature Engineering
  - Encode Sex feature values for male = 1 and female =0

```
# Convert male = 1, female=0
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
data_train['Sex'] = le.fit_transform(data_train['Sex'].astype(str))
data_train.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	1	22	1	0	A/5 21171	7.25	Unknown	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38	1	0	PC 17599	71.28	C85	C
2	3	1	3	Heikkinen, Miss. Laina	0	26	0	0	STON/O2. 3101282	7.92	Unknown	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35	1	0	113803	53.10	C123	S
4	5	0	3	Allen, Mr. William Henry	1	35	0	0	373450	8.05	Unknown	S

  - New Features

familySize feature refer to the passenger's family size. Hence, it is derived from sibling spouse (SibSp) and parent child (Parch) feature, and the formula is SibSp + Parch + 1

Isalone feature refer to whether the passenger is alone or not. Hence, it is derived from familySize, and it is calculated by if familySize is equal to 1, the passenger is alone.

A passenger traveling alone (isalone=1) may have had fewer social ties or support on board, which could have increased their vulnerability in a crisis situation like the sinking of the Titanic. However, a passenger with a large family size (familySize>1) may have had more people to look after or coordinate with during the evacuation, which could have increased their chances of survival.

- Normalization

```
# Feature Scaling, Normalizaiton
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

1.3.2 Modelling -

Supervised Learning – (Log Regression, SVC, RFC, CNN)

Logistic Regression with default setting

```
import time
start_time = time.time()
#Initialize, fit and predict
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(random_state=9)
logreg.fit(x_train_scaled, y_train)
y_pred = logreg.predict(x_test_scaled)
```

SVC with kernel RBF setting only

```
# SVC
from sklearn.svm import SVC
svc = SVC(kernel='rbf',random_state=9)
svc.fit(x_train_scaled_svc, y_train)
y_pred_svc = svc.predict(x_test_scaled_svc)
```

RFC with default setting

```
# Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(random_state=9)
rfc.fit(x_train_scaled, y_train_rfc)
y_pred_rfc = rfc.predict(x_test_scaled)
```

CNN settings (optimizer='adam', loss='binary_crossentropy', metric = 'accuarcy')

```
from tensorflow import keras

model = keras.Sequential([

    ## reshaping the input entries
    keras.layers.Dense(50, input_shape=(5,), activation='relu'),
    keras.layers.Dropout(0.50),     ## to avoid overfitting and underfiting

    ## creating the hidden layer
    keras.layers.Dense(100,activation='relu'),
    keras.layers.Dropout(0.70),     ##  to avoid overfitting and underfiting

    keras.layers.Dense(150,activation='relu'),
    keras.layers.Dropout(0.70),      ## to avoid overfitting and underfiting


    ## final neural layer
    keras.layers.Dense(1,activation='sigmoid')

])

model.compile(optimizer='adam',
              loss='binary_crossentropy',  ## since output in 0 or 1
              metrics=['accuracy'])

model.fit(x_train_scaled_cnn,y_train_cnn, epochs=100)

y_pred_cnn = model.predict(x_test_scaled_cnn)

cnn_score = model.evaluate(x_train_scaled_cnn,y_train_cnn)[1]
print("CNN Score:", cnn_score)
```

1.3.3 Modelling

Unsupervised Learning

```
from sklearn.cluster import MeanShift
ms = MeanShift(bandwidth= 30) #We will pr
#We found the bandwith using the estimat
ms.fit(data_train)
```

	Survived	Sex	Age	Fare	familySize	isalone	Counts
cluster_group							
0.0	0.321580	0.691114	27.686883	14.672539	1.710860	0.679831	709
1.0	0.581395	0.527132	35.162791	64.949922	2.705426	0.302326	129
2.0	0.757576	0.333333	32.818182	131.107576	2.393939	0.272727	33
3.0	0.647059	0.352941	31.117647	238.187059	3.058824	0.294118	17
4.0	1.000000	0.666667	35.333333	512.330000	1.333333	0.666667	3

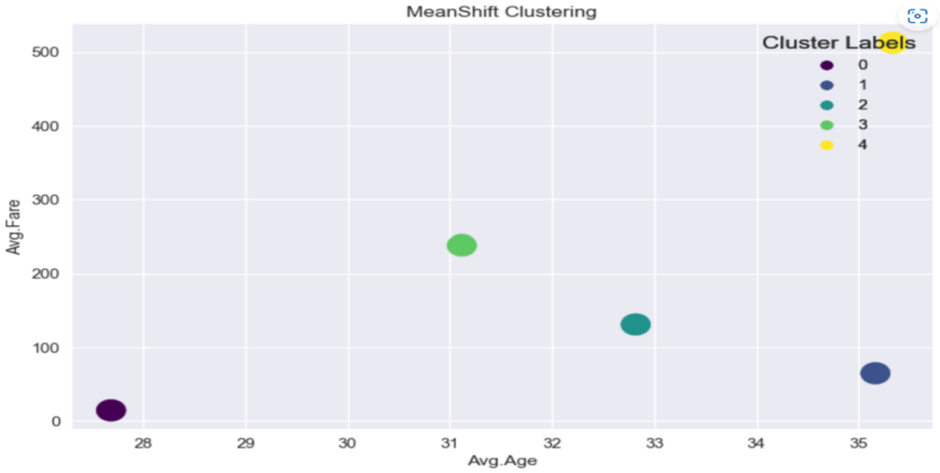
Cluster 0 i.e., the 1st Cluster
Have 709 passengers
Survival rate is 32% (very low) means most of them didn't survive
Mostly Male
Average family size of 1-2, 64% likely to be alone
The average fare paid is $14

Cluster 1 i.e., the 2nd Cluster
Have 129 passengers
Survival rate is 58% means most of them survived
Mostly Male
Average family size of 2-3, 30% likely to be alone
The average fare paid is $64

Cluster 2 i.e., the 3rd Cluster
Have 33 passengers
Survival rate is 75% means most of them survived
Mostly Female
Average family size of 2-3, 27% likely to be alone
The average fare paid is $131

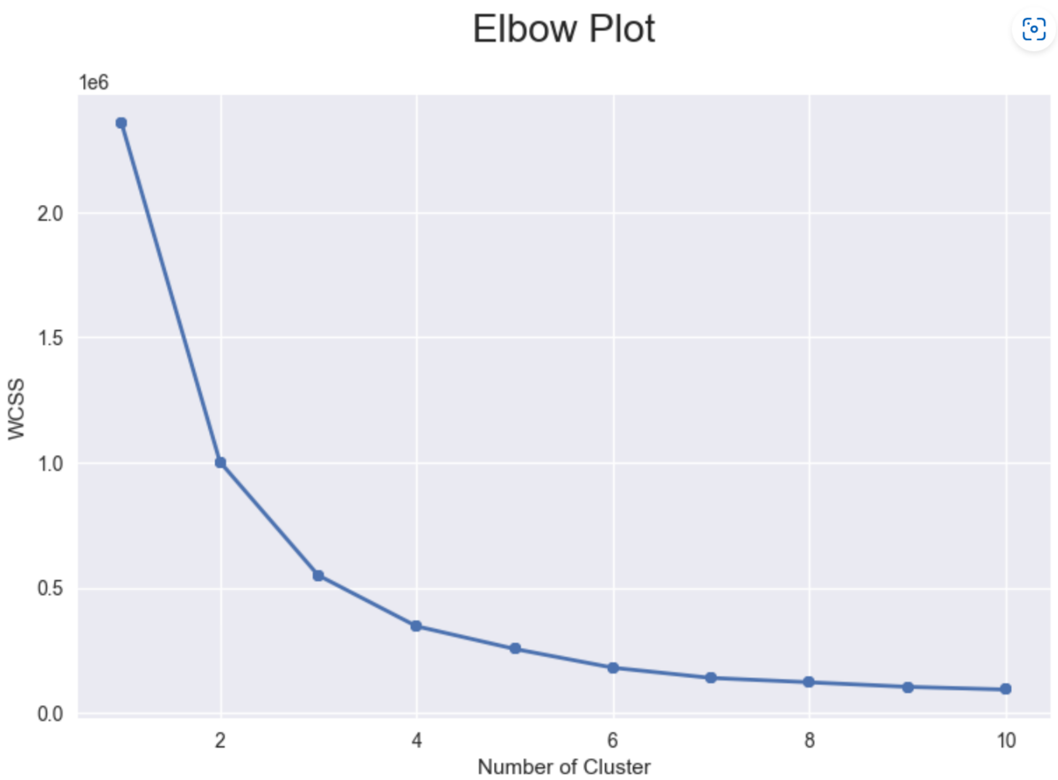
Cluster 3 i.e., the 4th Cluster
Have 17 passengers
Survival rate is 64% means most of them survived
Mostly Female
Average family size of 3-4, 29% likely to be alone
The average fare paid is $238

Cluster 4 i.e., the 5th Cluster
Have only 3 passengers (can be ignored, can consider as outliers as too little data)



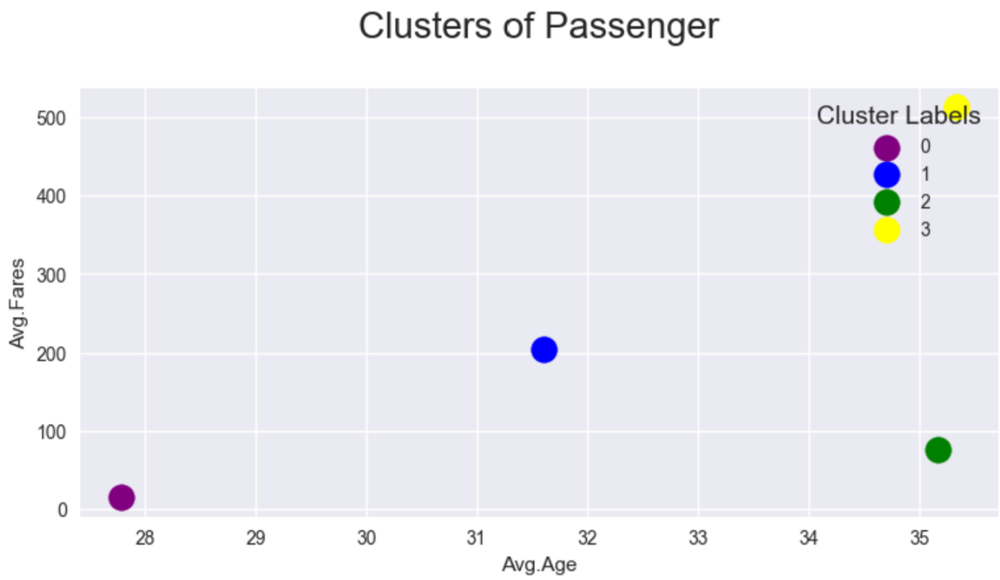
From the scatterplot above, we can see that there are 5 clusters group for average age by average fare. As compared to cluster 0 (first cluster), we can see that the rest of the clusters, average fares increases as the average age increase .

K-Means



With this elbow plot shown above, which tell us the best optimal numbers of clusters for k-means clustering. For this graph, we can see that the best optimal numbers of cluster are 4.

```
kmeans = KMeans(n_clusters = 4)
y_kmeans = kmeans.fit_predict(X)
print(y_kmeans)
print(kmeans.cluster_centers_)
```



With the help of an elbow plot, we can determine the optimal number of clusters, and in this case, we have predefined 4 clusters. The clustering pattern is like mean shift clustering, which can handle complex cluster shapes and adaptively determine the number of clusters. However, unlike mean shift, k-means clustering may not be able to identify outliers effectively. This limitation can potentially affect the accuracy of the clustering results as outliers might be incorrectly assigned to clusters.

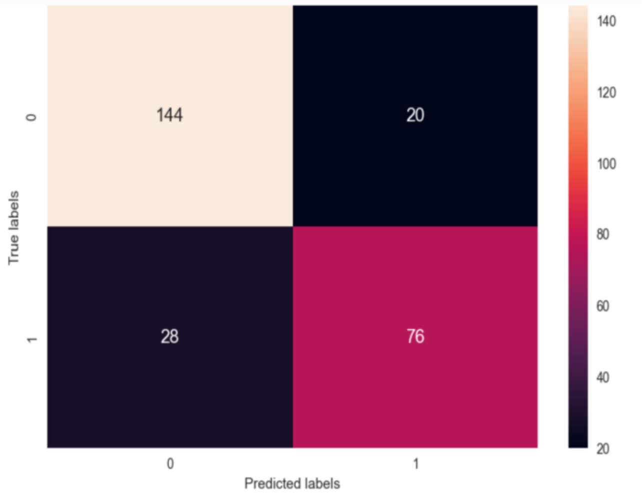
1.3.4 Evaluation

- Supervised Learning - Classification

```
Logistic Regression Model
Train set score: 0.79
Test set score: 0.82
Elapsed time: 0.14 seconds

----------------------------

Evaluation
Accuarcy Score is: 0.82
Precision Score is :  0.79
Recall Score is :  0.73
Confusion Matrix
[[144  20]
 [ 28  76]]
```

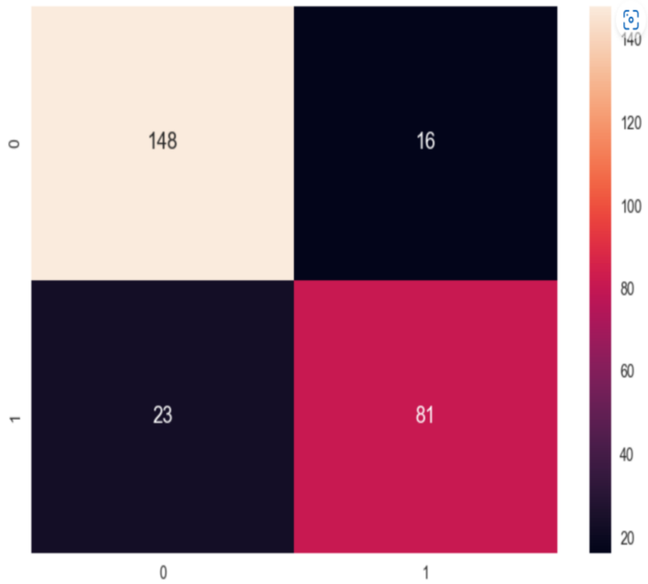


The logistic regression model, train and test score is 0.79 and 0.82 respectively. This suggests that the model is not overfitting the training data, as it is able to generalize well to new, unseen data. and it took 0.14s to run. The RFC had an accuracy score of 0.82. The confusion matrix indicates that it had predicted 220(144+76) correct passenger survival.

```
SVC-RBF Model
Train set score: 0.820
Test set score: 0.854
Elapsed time: 0.06 seconds

----------------------------

Evaluation
Accuarcy Score is: 0.85
Precision Score is :  0.84
Recall Score is :  0.78
Confusion Matrix
[[148  16]
 [ 23  81]]
```

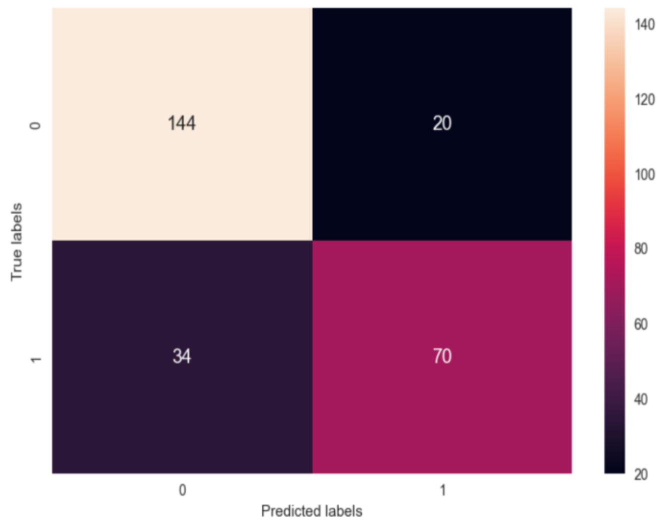


The SVC-RBF model, train and test score is 0.82 and 0.85 respectively. This suggests that the model is not overfitting the training data, as it is able to generalize well to new, unseen data. and it took 0.06s to run. The RFC had an accuracy score of 0.85. The confusion matrix indicates that it had predicted 229(148+81) correct passenger survival.

```
RFC Model
RFC Train set score: 0.99
RFC Test set score: 0.80
Elapsed time: 0.41 seconds

----------------------------

Evaluation
Accuarcy Score is: 0.8
Precision Score is :  0.78
Recall Score is :  0.67
Confusion Matrix
[[144  20]
 [ 34  70]]
```

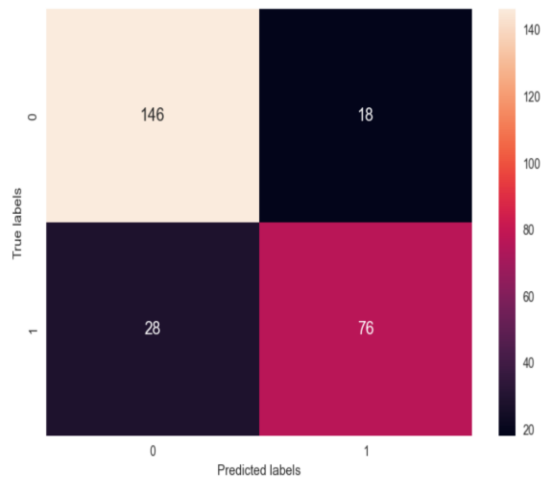


The RFC model, train and test score is 0.99 and 0.80 respectively. This shows that the data is overfitted and it took 0.41s to run. RFC had an accuracy score of 0.80. The RFC confusion matrix indicates that it had predicated 214(144+70) correct passenger survival.

```
CNN Model
CNN Train set score: 0.99
CNN Test set score: 0.80
Elapsed time: 18.02 seconds

----------------------------

Evaluation
Accuracy: 0.83
Confusion Matrix
[[146  18]
 [ 28  76]]
```



The CNN model, train and test score is 0.99 and 0.80 respectively. This shows that the data is overfitted and it took 18.02s to run. The accuracy score of CNN is 0.83. The CNN confusion matrix indicates that it had predicated 222(146+76) correct passenger survival.

- ROC_AUC

Models	AUC value
Logistic Regression	0.80
SVC-RBF	0.84
RFC	0.78
CNN	0.89

The area under the ROC curve (AUC) is a commonly used metric to summarize the overall performance of the model. The AUC value ranges from 0 to 1, where 0.5 corresponds to random guessing and 1 corresponds to a perfect classifier.

All 4 models had a AUC value of 0.80 – 0.89 which suggests that all 4 models has good discriminatory power in distinguishing between the positive and negative class labels. Specifically, the model's ability to correctly classify positive samples (sensitivity) and its ability to avoid false alarms (1-specificity) are reasonably balanced, resulting in a relatively high AUC value.

- Feature Importance

Models	Features (Descending Order)
Logistic Regression	Sex: 0.2456 familySize: 0.0369 Age: 0.0257 isalone: 0.0146 Fare: 0.0127
SVC-RBF	Sex: 0.2456 familySize: 0.0369 Age: 0.0257 isalone: 0.0146 Fare: 0.0127
RFC	Fare: 0.3597 Sex: 0.2750 Age: 0.2601 familySize: 0.0898 isalone: 0.0154

The feature importance values provided for each model indicate the relative contribution of each feature in predicting whether the passenger can survive or not on the Titanic. The higher the value of the feature importance, the more important the feature is in making accurate predictions.

In the Logistic Regression and SVC-RBF models, the top five features by importance are: Sex, familySize, Age, isalone, and Fare. However, all five features have relatively low importance values, indicating that none of them are particularly strong predictors of survival.

In contrast, the Random Forest Classifier (RFC) model assigns much higher importance values to the top four features: Fare, Sex, Age, and familySize. This suggests that these features play a more important role in determining whether a passenger survived or not.

1.3.5 Model Optimization (GridsearchCV)

Model	Best hyparameters	Best Score	Before Optimization Accuracy	After Optimization Accuracy
Logistic Regression	{'logisticregression__C': 10, 'logisticregression_max_iter': 100, 'logisticregression__penalty': 'l1', 'logisticregression__solver': 'liblinear'}	0.78	0.82	0.81
SVC	{'C': 1, 'gamma': 'scale'}	0.81	0.85	0.85
RFC	{'max_depth': 5, 'max_features': 'sqrt', 'min_samples_split': 10, 'n_estimators': 200}	0.80	0.80	0.87
CNN	{'activation': 'relu', 'batch_size': 16, 'dropout_rate': 0.2, 'epochs': 100, 'num_units': 128, 'optimizer': 'adam'}	0.82	0.83	0.84

Table for Supervised Learning: model optimization with gridsearchCV

After model optimization with gridsearchCV, we can find the best hyperparameters for each model to get better result. As the table above, RFC has increased the most accuracy from 0.80 to 0.87.

Unsupervised Learning (Optimization)

Mean shift clustering

```
from sklearn.metrics import silhouette_score
# Range of bandwidth values to explore
bandwidths = [10.0, 25.0,35.0,50.0,74.0]

# For each bandwidth value, fit Mean Shift clustering and compute the Silhouette score
for bandwidth in bandwidths:
    ms = MeanShift(bandwidth=bandwidth, bin_seeding=True)
    ms.fit(data_train)
    labels = ms.labels_
    if len(set(labels)) > 1:
        silhouette_avg = silhouette_score(data_train, labels)
        print("For bandwidth =", bandwidth, "the average silhouette score is :", silhouette_avg)
    else:
        print("For bandwidth =", bandwidth, "the number of clusters is 1")
# silhouette score >0.5 a strong indication of good clustering
# silhouette score <0.2 considered weak and may indicate that the data does not have a clear clustering structure.
```

```
For bandwidth = 10.0 the average silhouette score is : 0.42740419579681155
For bandwidth = 25.0 the average silhouette score is : 0.5910755143013338
For bandwidth = 35.0 the average silhouette score is : 0.682005611582051
For bandwidth = 50.0 the average silhouette score is : 0.7917491497717523
For bandwidth = 74.0 the average silhouette score is : 0.7917491497717523
```

For a silhouette score greater than 0.5 is a strong indication of good clustering. In this case, I choose bandwidth = 35 which had a silhouette score of 0.68.

	Survived	Sex	Age	Fare	familySize	isalone	Counts
cluster_group							
0.0	0.338422	0.681934	28.278626	18.538053	1.858779	0.642494	786
1.0	0.729412	0.388235	35.552941	100.435412	2.117647	0.294118	85
2.0	0.647059	0.352941	31.117647	238.187059	3.058824	0.294118	17
3.0	1.000000	0.666667	35.333333	512.330000	1.333333	0.666667	3

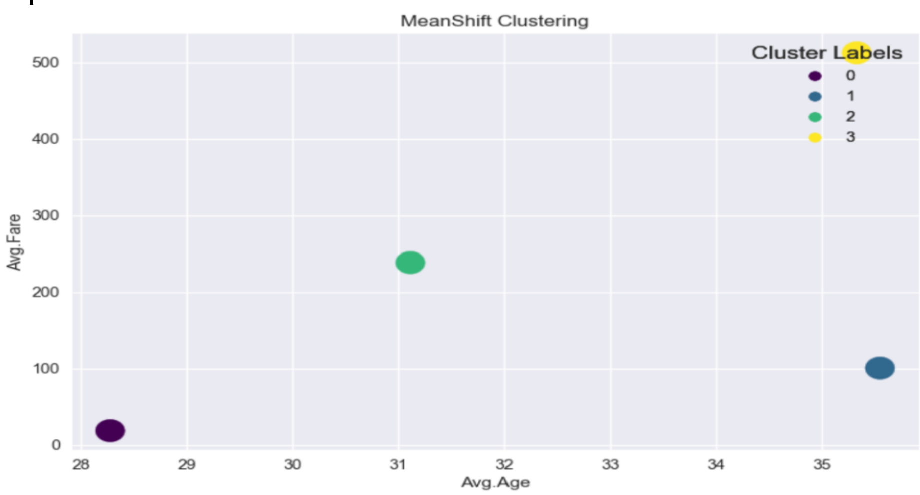
Cluster 0 i.e., the 1st Cluster
Have 786 passengers
Survival rate is 33% (very low) means most of them didn't survive
Mostly Male
Average family size of 1-2, 64% likely to be alone
The average fare paid is $18

Cluster 1 i.e., the 2nd Cluster
Have 85 passengers
Survival rate is 72% means most of them survived
Mostly Female
Average family size of 2-3, 29% likely to be alone
The average fare paid is $100

Cluster 2 i.e., the 3rd Cluster
Have 17 passengers
Survival rate is 64% means most of them survived
Mostly Female
Average family size of 3-4, 29% likely to be alone
The average fare paid is $238 (which is far higher than the 1st cluster average fare)

Cluster 3 i.e., the 4th Cluster
Have only 3 passengers (can be ignored, can consider as outliers as too little data)

For Cluster 1 and 2, they are most likely not to be alone and they have higher chance of survival. As compared to Cluster 0, they are most likely to be alone and lower chance of survival. Thus, we can conclude that passenger that survived are most likely to be Female with at least 1 family member travelling with them and they paid a much higher fares as compared to cluster 0.



After getting the silhouette score optimization for mean shift clustering, I found the most optimal bandwidth around 35 to 74 which help to reduce clustering from 5 to 4.

3.0 Discussion and Conclusions

- Summary of project achievements

➢ Titanic dataset

For supervised models (Logistic Regression, SVC-RBF, RFC & CNN), I will choose RFC as the ideal model. After gridsearchCV optimization, I can find the best hyperparameters for all the 4 different models and the accuracy score that increased the most from 0.80 to 0.87 is RFC.

RFC can handle non-linear relationships between features and the target variable. It can capture complex interactions and non-linear decision boundaries, unlike logistic regression, which assumes linear relationships.

RFC is robust to outliers and noisy data. Outliers have minimal impact on the overall performance of the model since each decision tree in the forest is less likely to be influenced by a single outlier. Logistic regression and SVM with an RBF kernel can be sensitive to outliers, affecting their performance. CNNs excel in image and pattern recognition tasks but require a large amount of labeled data and computational resources.

For unsupervised models (Mean shift clustering, K-means clustering), I will choose Mean Shift as it does not require specifying the number of clusters in advance. It automatically determines the number of clusters based on the data distribution, making it suitable for scenarios where the number of clusters is unknown or variable. In contrast, K-means requires the number of clusters to be predefined.

Mean Shift clustering is relatively robust to noise and outliers in the data. By estimating the density gradient, it can identify cluster centers even in the presence of noisy or outlying data points. K-means, on the other hand, is sensitive to outliers and can be influenced by their presence, potentially affecting the quality of the clustering results.

Mean Shift is advantageous when the number of clusters is unknown, clusters have complex shapes, or noise/outliers are present. On the other hand, K-means is advantageous when computational efficiency, interpretability, and scalability are important considerations.

- Future Direction for improvement

We considered the four options below for future improvements. To begin, we can communicate more with business users to better understand their business concepts and, hopefully, improve model prediction performance by leveraging their experience. Second, we can use improved techniques to continue fine-tuning the parameters. Third, we can search for more efficient algorithms to run the datasets.

Lastly, we can try to gather additional data to augment existing dataset. More data can help increase the diversity and representation of our samples, improving the model's generalization capabilities. If the data test set is small, it may not adequately represent the entire population or the true distribution of the data. The results may be more prone to random variations and may not provide a reliable estimate of model performance.