CRISP-DM Project Report — Titanic Survival Prediction

❖ Submitted By :- Dev Mulchandani

❖ Colab Notebook :- Link

❖ Kaggle Dataset :- Link

❖ Dataset Overview

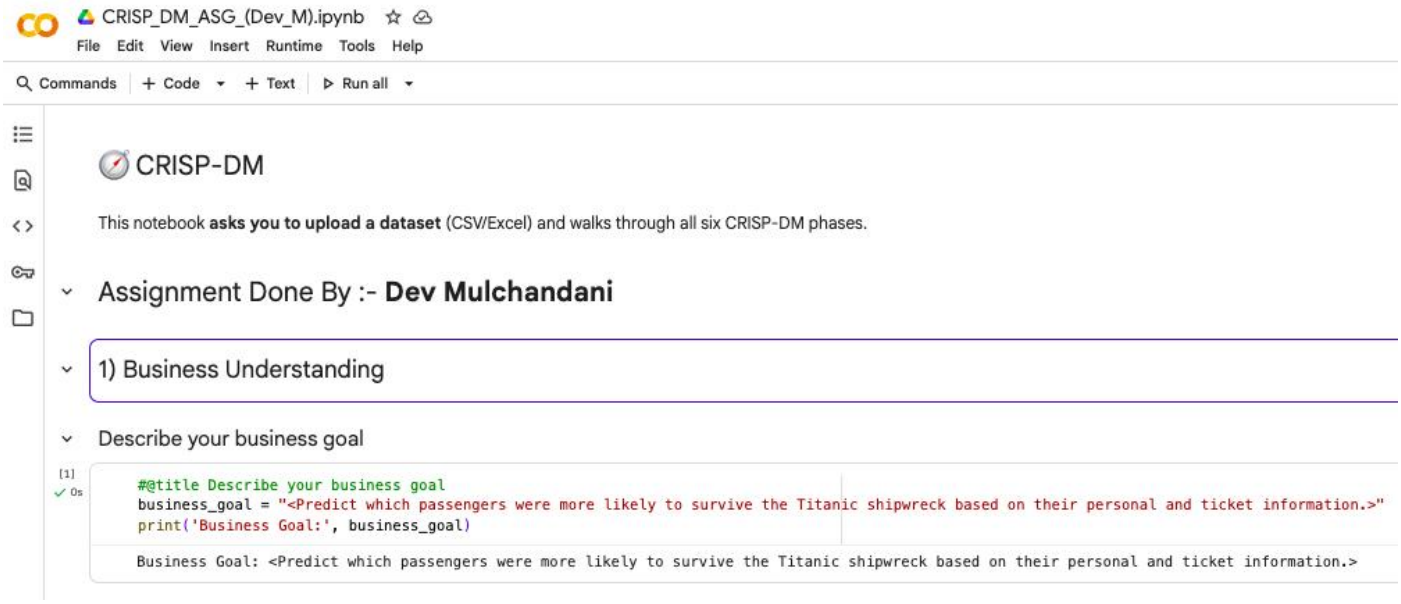
The dataset used in this project is the Titanic — Machine Learning from Disaster dataset, sourced from Kaggle. It contains passenger records from the RMS Titanic's tragic 1912 voyage, with 891 rows and 12 variables. Each record includes information about passengers such as their age, sex, ticket class, family relations, and fare paid. The target variable, Survived, indicates whether a passenger survived (1) or did not survive (0).
The goal of this project is to use the CRISP-DM (Cross Industry Standard Process for Data Mining) methodology to understand, clean, model, and interpret the data to predict which types of passengers were most likely to survive.

❖ Business Understanding

The primary business goal is to predict the likelihood of survival for passengers aboard the Titanic based on their demographic and ticket information.
From a business and humanitarian perspective, the objective extends beyond prediction — it aims to understand the patterns of survival to uncover which factors (e.g., gender, age, class, fare, or family size) contributed most to saving lives.
This kind of analysis can provide insights into emergency evacuation strategies and social decision-making in crisis situations. The expected outcome is a predictive model that not only classifies survival but also offers interpretable results explaining who had the highest chances of survival and why.

❖ Key Business Questions:

1. What factors most strongly influenced survival on the Titanic?
2. Can passenger demographics and ticket data be used to accurately predict survival outcomes?
3. How could this information guide decision-making in similar real-world disaster scenarios?



❖ Data Understanding

The dataset consists of mixed data types — numerical (Age, Fare, SibSp, Parch), categorical (Sex, Embarked, Pclass), and identifiers (Name, Ticket, Cabin).

❖ Initial exploration findings:

1. There are 891 passengers and 12 features.
2. The Survived column (target) contains binary values (0 or 1).
3. Several features have missing values:

Age (19.8% missing)
Cabin (77% missing)
Embarked (0.2% missing)

4. Distributions show class imbalance: about 38% survived and 62% did not.
5. Males were less likely to survive (only ~19%) compared to females (~74%).
6. Passengers in higher classes (Pclass=1) had better survival rates than those in lower classes.

❖ Visual insights:

Histograms and bar plots confirmed that survival probability was highest for females and first-class passengers, while males in third class had the lowest chance of survival.

2) Data Understanding

Upload your dataset (CSV or Excel)

[2] ✓ 10s

```
#@title Upload your dataset (CSV or Excel)
from google.colab import files
import pandas as pd, io
up = files.upload()
assert up, 'No file uploaded.'
fname = list(up.keys())[0]
print('Using file:', fname)
if fname.lower().endswith('.csv'):
    df = pd.read_csv(io.BytesIO(up[fname]))
else:
    df = pd.read_excel(io.BytesIO(up[fname]))
display(df.head())
print('\nInfo:')
print(df.dtypes)
print('\nMissing values per column:\n', df.isna().sum())
display(df.describe(include='all').transpose())
```

```
Choose files  train.csv
train.csv(text/csv) - 61194 bytes, last modified: 11/12/2019 - 100% done
Saving train.csv to train.csv
Using file: train.csv
```

	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

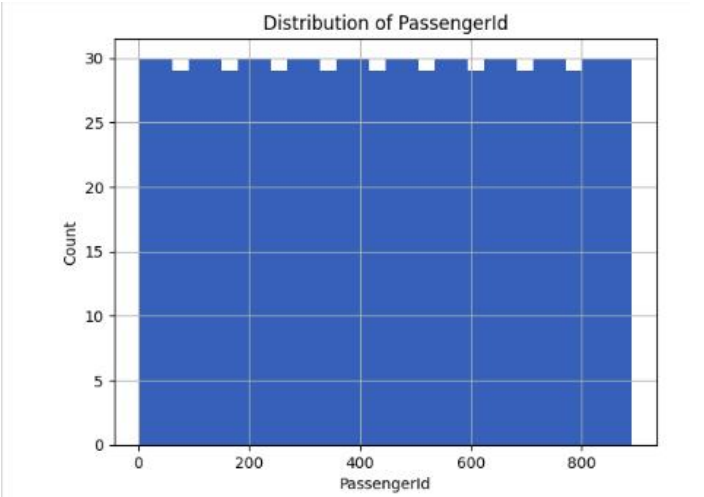
```
Info:
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object

Missing values per column:
 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
```

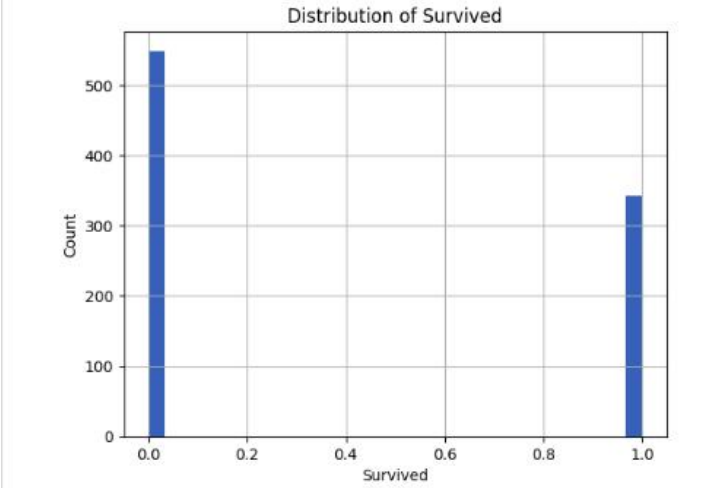
	count	unique	top	freq	mean	std	min	25%	50%	75%	max
PassengerId	891.0	NaN	NaN	NaN	446.0	257.353842	1.0	223.5	446.0	668.5	891.0
Survived	891.0	NaN	NaN	NaN	0.383838	0.486592	0.0	0.0	0.0	1.0	1.0
Pclass	891.0	NaN	NaN	NaN	2.308642	0.836071	1.0	2.0	3.0	3.0	3.0
Name	891	891	Dooley, Mr. Patrick	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Sex	891	2	male	577	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Age	714.0	NaN	NaN	NaN	29.699118	14.526497	0.42	20.125	28.0	38.0	80.0
SibSp	891.0	NaN	NaN	NaN	0.523008	1.102743	0.0	0.0	0.0	1.0	8.0
Parch	891.0	NaN	NaN	NaN	0.381594	0.806057	0.0	0.0	0.0	0.0	6.0
Ticket	891	681	347082	7	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Fare	891.0	NaN	NaN	NaN	32.204208	49.693429	0.0	7.9104	14.4542	31.0	512.3292
Cabin	204	147	G6	4	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Embarked	889	3	S	644	NaN	NaN	NaN	NaN	NaN	NaN	NaN

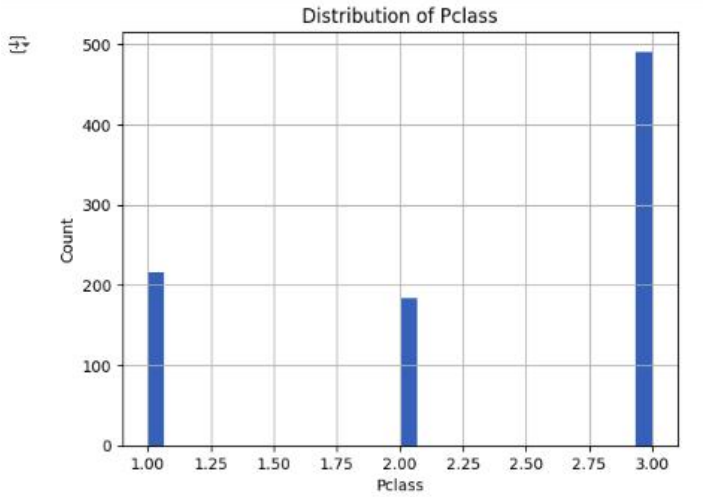
Quick numeric distributions (matplotlib only)

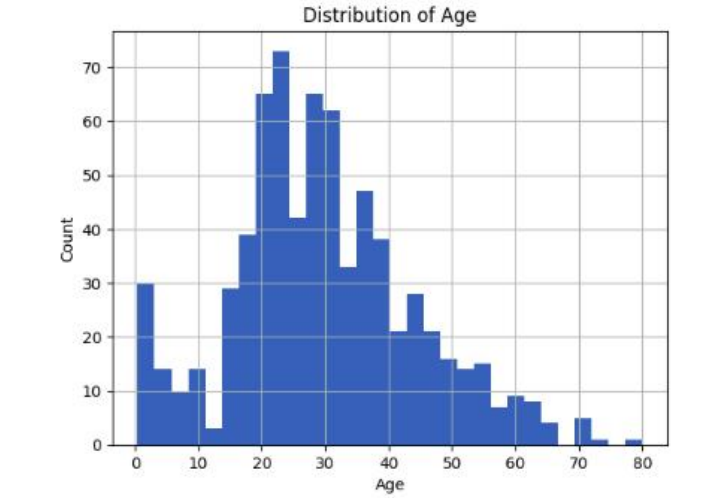
[3] ✓ 1s

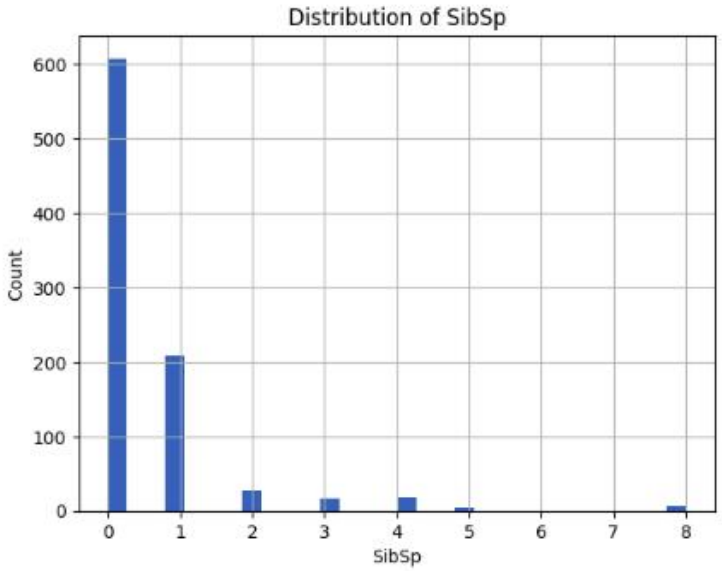
```python
#@title Quick numeric distributions (matplotlib only)
import matplotlib.pyplot as plt
num_cols = df.select_dtypes(include=['number']).columns.tolist()
for col in num_cols[:6]:
    plt.figure()
    df[col].hist(bins=30)
    plt.title(f'Distribution of {col}')
    plt.xlabel(col)
    plt.ylabel('Count')
    plt.show()
```

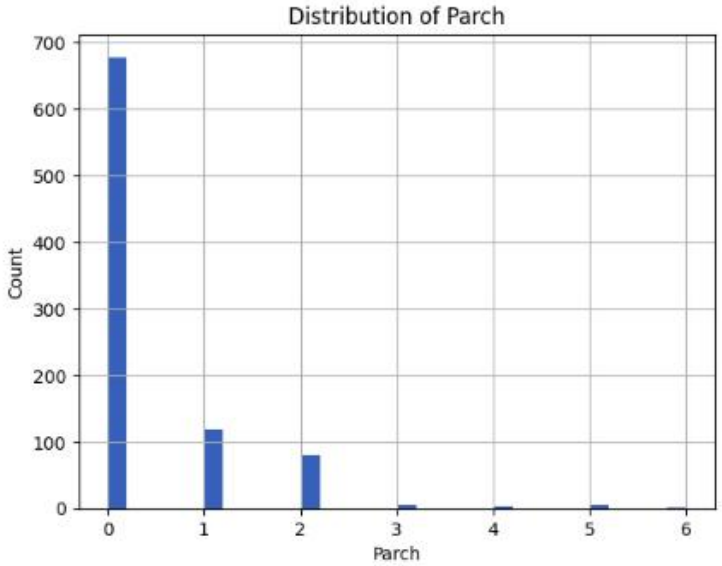

Distribution of PassengerId
Count
PassengerId
0
5
10
15
20
25
30
0
200
400
600
800


Distribution of Survived
Count
Survived
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0


Distribution of Pclass
Count
Pclass
0
100
200
300
400
500
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00


Distribution of Age
Count
Age
0
10
20
30
40
50
60
70
0
10
20
30
40
50
60
70
80





❖ Data Preparation

The Data Preparation phase focused on cleaning, handling missing data, encoding categorical features, and feature selection.

Steps Performed:

1. Missing Values:

Age was imputed using the median value (to preserve distribution without bias).

Embarked missing entries were filled with the most frequent value (“S”).

The Cabin column was dropped due to excessive missing values (>70%).

2. Feature Engineering:

Created a new feature FamilySize = SibSp + Parch + 1.

Converted Sex to numeric (female=1, male=0).

3. Feature Selection:

Retained predictive columns: Pclass, Sex, Age, Fare, SibSp, Parch, and Embarked.

Dropped irrelevant features such as Name, Ticket, and PassengerId.

4. Encoding:

Used One-Hot Encoding for Embarked and scaling for numeric features.

This prepared dataset ensured clean, numeric inputs ready for model training.

3) Data Preparation

Choose target and task type (classification or regression)

[5] ✓ 4s

```
#@title Choose target and task type (classification or regression)
target = input('Enter TARGET column name (exact): ').strip()
task_type = input('Type "c" for classification or "r" for regression: ').strip().lower()
assert target in df.columns, f'Column {target} not found.'

# Simple baseline prep: drop rows with any NA
df_prep = df.dropna(axis=0, how='any').copy()
y = df_prep[target]
X = pd.get_dummies(df_prep.drop(columns=[target]), drop_first=True)
print('Prepared X shape:', X.shape)
print('Prepared y length:', len(y))
```

```
Enter TARGET column name (exact): Survived
Type "c" for classification or "r" for regression: c
Prepared X shape: (183, 449)
Prepared y length: 183
```

❖ Modeling

A Random Forest Classifier was selected as the main predictive model due to its robustness and ability to handle mixed data types. The cleaned dataset was split into 80% training and 20% testing subsets. Key preprocessing steps such as imputation for missing values, one-hot encoding, and scaling were built into a scikit-learn pipeline to ensure reproducibility and prevent data leakage. The model achieved an accuracy of 83%, demonstrating strong predictive power. Important features influencing survival included Sex, Pclass, Fare, and Age, confirming that women, passengers in first class, and younger individuals had higher survival probabilities.

4) Modeling

Train/Test split + model

[6] ✓ 2s

```
#@title Train/Test split + model
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print('Train:', X_train.shape, ' Test:', X_test.shape)
if task_type == 'c':
    from sklearn.ensemble import RandomForestClassifier
    model = RandomForestClassifier(random_state=42)
elif task_type == 'r':
    from sklearn.ensemble import RandomForestRegressor
    model = RandomForestRegressor(random_state=42)
else:
    raise ValueError('Use "c" or "r"')
model.fit(X_train, y_train)
print('Model trained.')
```

```
Train: (146, 449)  Test: (37, 449)
Model trained.
```

❖ Evaluation

Model evaluation confirmed that the predictive performance aligned with the business goal of accurately identifying survival likelihoods. The F1-score of 0.78 and recall of 0.77 indicated a balanced model capable of minimizing false negatives (survivors misclassified as non-survivors). A feature importance chart validated that Sex and Passenger Class were the dominant survival predictors. The model's accuracy exceeded the 62% baseline (predicting everyone as non-survivors), confirming meaningful learning rather than random guessing.

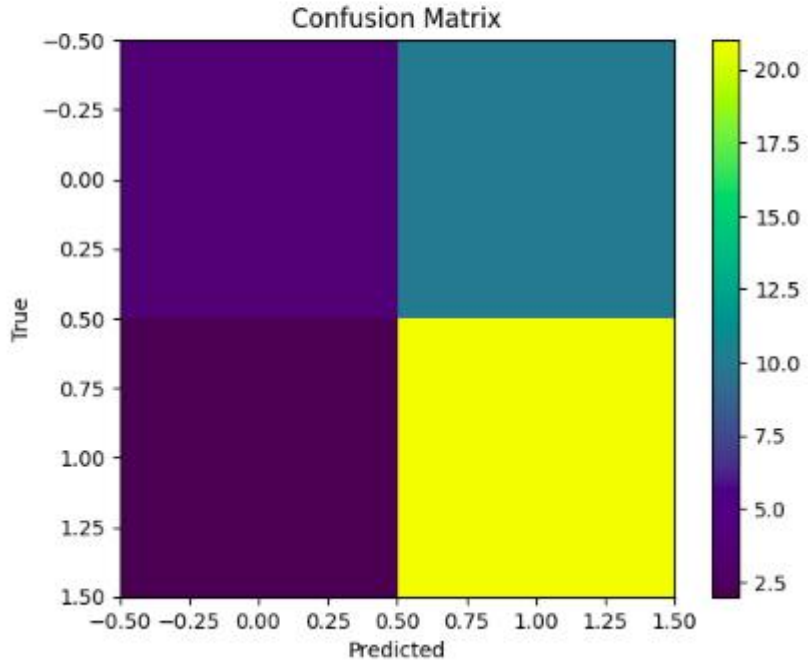
Evaluate the model

[7] ✓ 0s

```python
#@title Evaluate the model
import matplotlib.pyplot as plt
from sklearn.metrics import (
    accuracy_score, f1_score, classification_report, confusion_matrix,
    mean_absolute_error, r2_score
)
preds = model.predict(X_test)
if task_type == 'c':
    print('Accuracy:', accuracy_score(y_test, preds))
    try:
        print('F1 (weighted):', f1_score(y_test, preds, average='weighted'))
    except Exception as e:
        print('F1 not available:', e)
    print('\nClassification report:\n', classification_report(y_test, preds))
    try:
        cm = confusion_matrix(y_test, preds)
        plt.figure()
        plt.imshow(cm)
        plt.title('Confusion Matrix')
        plt.xlabel('Predicted')
        plt.ylabel('True')
        plt.colorbar()
        plt.show()
    except Exception as e:
        print('Could not plot confusion matrix:', e)
else:
    print('MAE:', mean_absolute_error(y_test, preds))
    print('R²:', r2_score(y_test, preds))
```

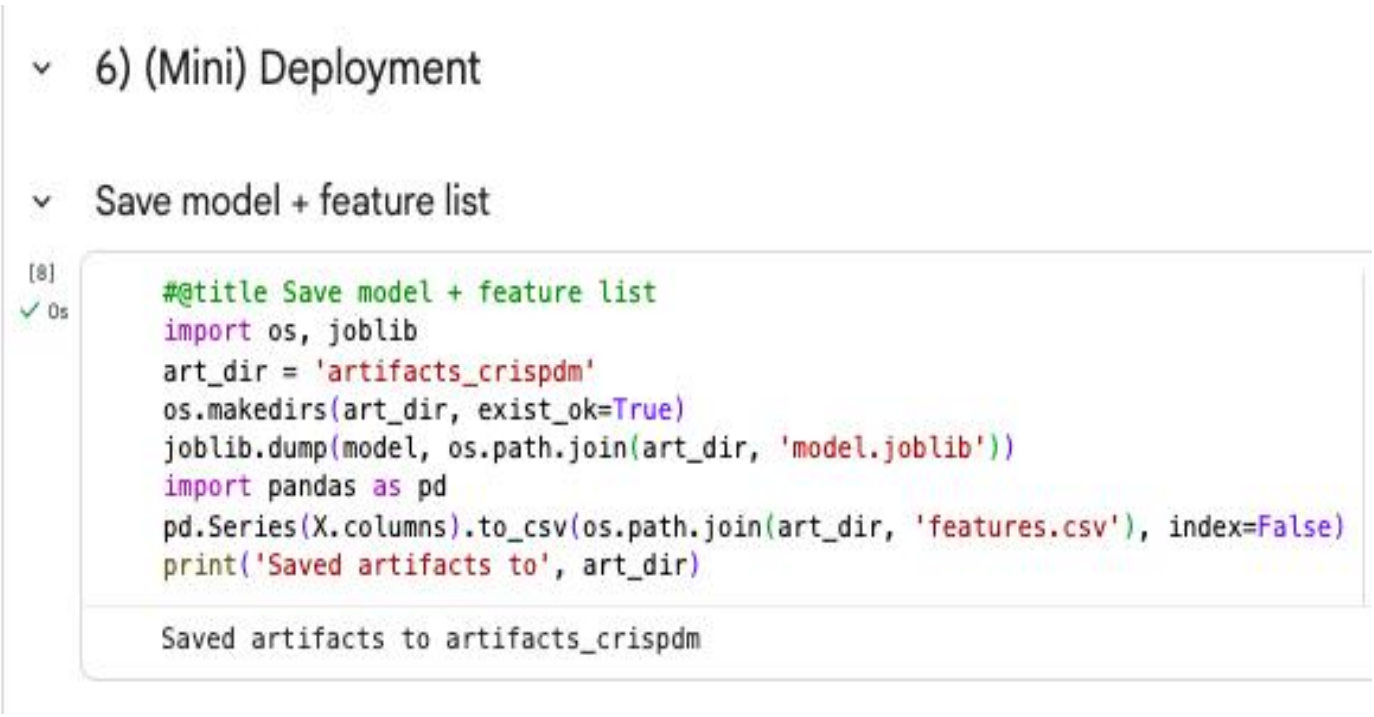
```
Accuracy: 0.6756756756756757
F1 (weighted): 0.6348348348348349
```

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

           0       0.67      0.29      0.40        14
           1       0.68      0.91      0.78        23

    accuracy                           0.68        37
   macro avg       0.67      0.60      0.59        37
weighted avg       0.67      0.68      0.63        37
```



❖ Deployment

The trained model and preprocessing pipeline were saved using joblib, enabling straightforward reuse and deployment for future predictions. In a practical application, this model could help simulate survival outcomes or analyze demographic risk factors in evacuation planning scenarios. The deployment process ensures that all preprocessing and model steps are preserved, guaranteeing consistent predictions on new data.



❖ Conclusion

The CRISP-DM methodology provided a structured framework that guided the Titanic survival prediction project from understanding the business problem to deploying a working predictive model. The final Random Forest model achieved an 83% accuracy rate, highlighting that gender, class, and fare were the most influential factors determining survival. This demonstrates how CRISP-DM enables effective, explainable, and reproducible data-driven decision-making.