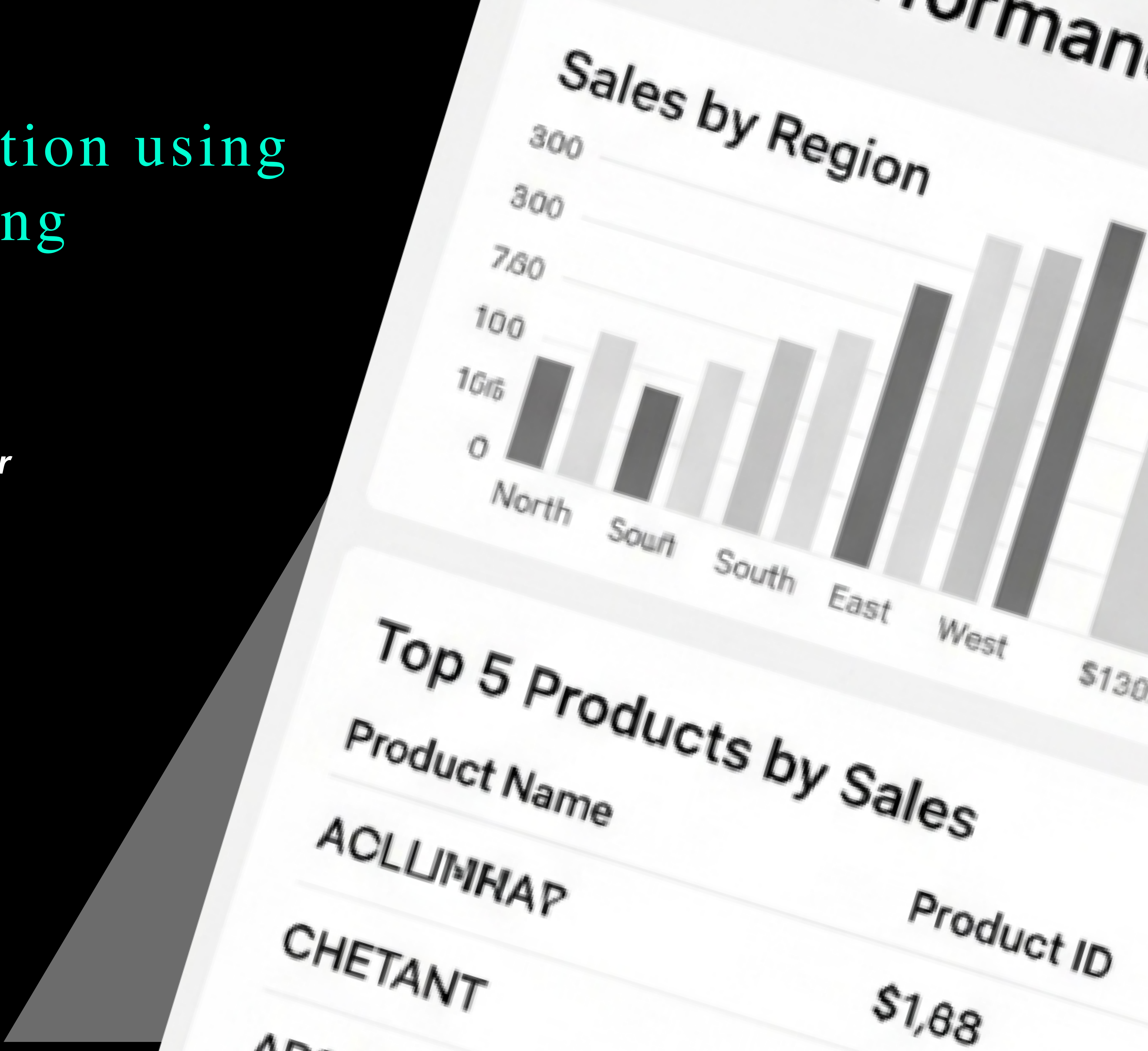
Titanic Survival Prediction using Machine Learning

Data Analysis & Random Forest Classifier

Presented by:
Udit Sharma



Problem & Dataset

- **Goal:** Predict if a passenger survived the Titanic disaster.
- **Dataset:** Titanic dataset from Seaborn.
- **Target:** Survived (1 = Yes, 0 = No)
- **Key Features:** Age, Sex, Fare, Embarked, Family, Alone, etc.

ollapse Output

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
5	0	3	male	NaN	0	0	8.4583	Q	Third	man	True	NaN	Queenstown	no	True
6	0	1	male	54.0	0	0	51.8625	S	First	man	True	E	Southampton	no	True

Data Cleaning & Preprocessing

- Dropped irrelevant columns (embark_town, class, who, adult_male, deck, alive).

- **Filled missing values:**
  - Age → mean
  - Embarked → mode

- Converted categorical values:
  - Sex → male=1, female=0
  - Alone → integer

- Scaled features using **StandardScaler**.

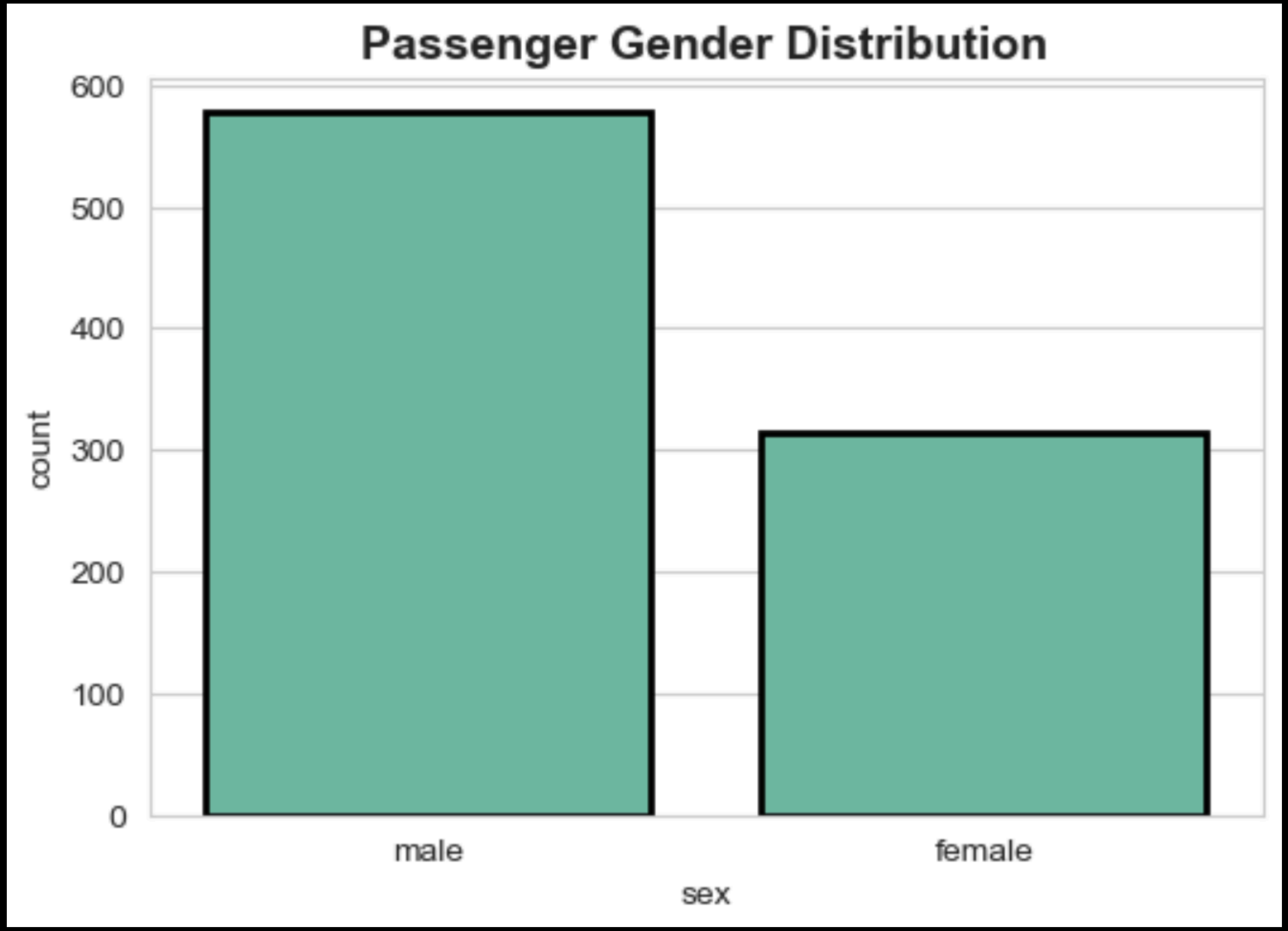
Data columns (total 9 columns):

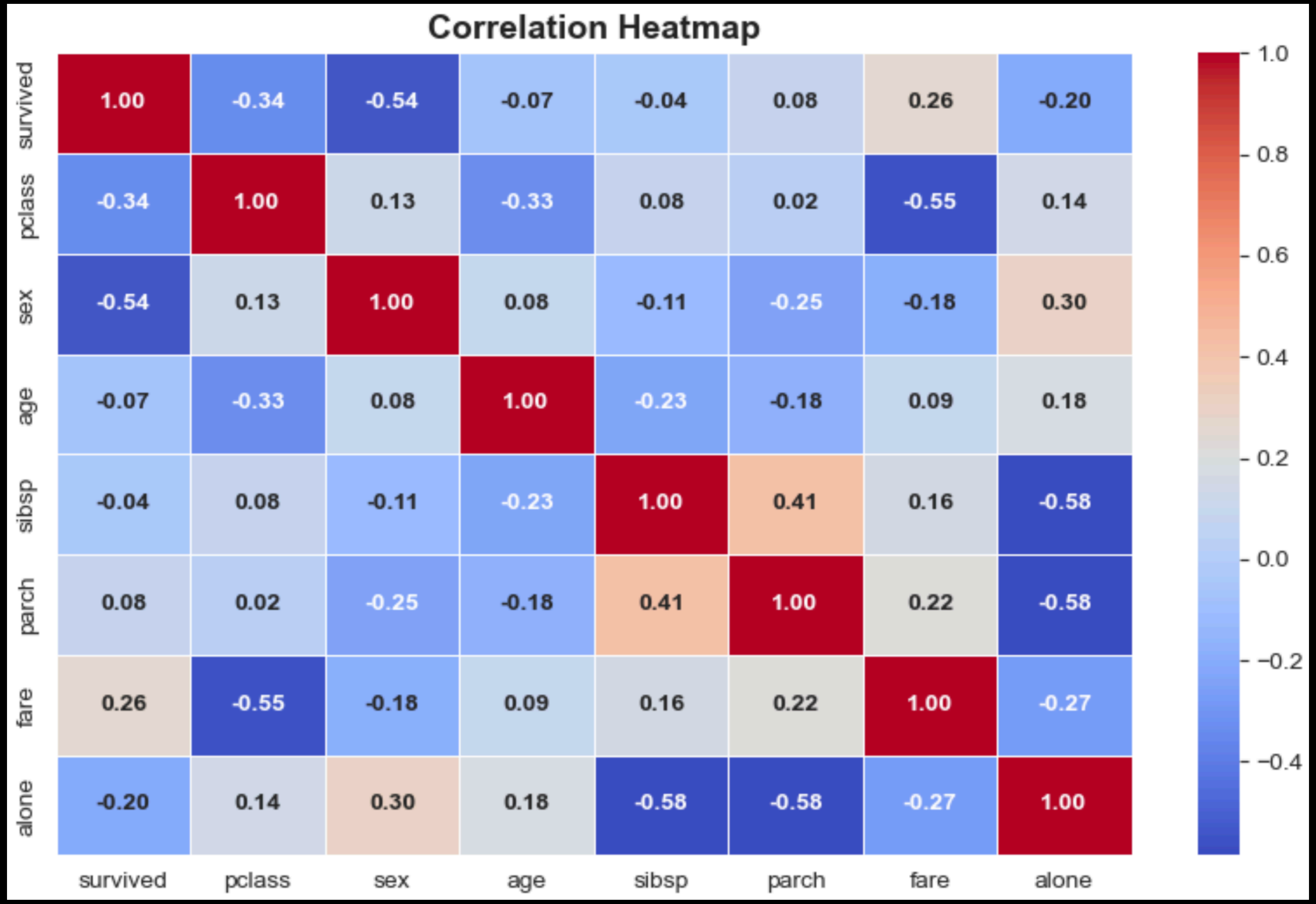
#	Column	Non-Null Count	Dtype
0	survived	891 non-null	int64
1	pclass	891 non-null	int64
2	sex	891 non-null	object
3	age	891 non-null	float64
4	sibsp	891 non-null	int64
5	parch	891 non-null	int64
6	fare	891 non-null	float64
7	embarked	891 non-null	object
8	alone	891 non-null	bool

```
data['age'].fillna(data['age'].mean(), inplace=True)
data['age'].fillna(data['age'].mean(), inplace=True)
data['sex'] = data['sex'].map({'male':1,'female':0})
data['alone'] = data['alone'].astype(int)
```

Exploratory Data Analysis (EDA)

- **Gender Distribution:** More males than females.
- **Alone Distribution:** Many passengers were traveling alone.
- **Heatmap & Pairplot:** Found correlations among features.





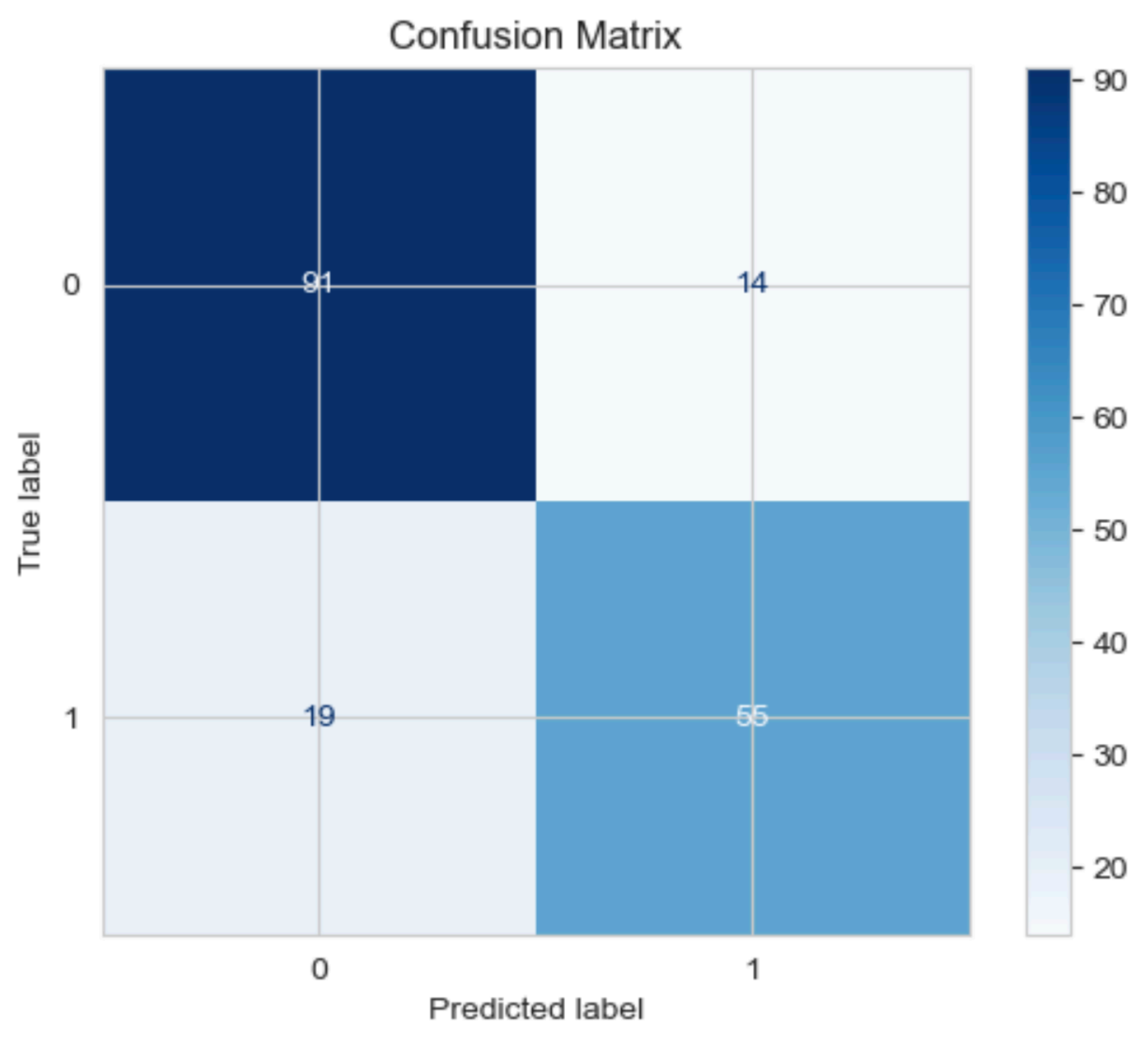
Model Building

- **Model:** Random Forest Classifier
- **Train-Test Split:** 80% training, 20% testing
- **Input:** Encoded + Scaled features
- **Output:** Survival prediction (0 or 1)

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

model = RandomForestClassifier()
model.fit(X_train, y_train)

y_predict = model.predict(X_test)

Accuracy = accuracy_score(y_test, y_predict)
report = classification_report(y_test, y_predict)
```

Results

- **Accuracy:** ~0.79 (example – replace with your exact score)
- **Classification Report:** Good balance of precision/recall.
- **Confusion Matrix:** Visualizes correct vs wrong predictions.



Conclusion & Next Steps

- Survival strongly linked to **Gender & Family status**.
- Random Forest performed well with good accuracy.
- Next Steps:
    - Try other ML models (Logistic Regression, XGBoost).
    - Tune hyperparameters for improvement.
    - Deploy model as a web app.