# TITANIC - Machine Learning from Disaster


*N M A Chamindu Nipun*

IEEE
Advancing Technology for Humanity

# Titanic - Machine Learning from Disaster



Titanic disaster is one of the most famous shipwrecks in the world history.
Titanic was a British cruise liner that sank in the North Atlantic Ocean a few hours after colliding with an iceberg.
While there are facts available to support the cause of the shipwreck, there are various speculations regarding the survival rate of passengers in the Titanic disaster.
Over the years, data of survived as well as deceased passengers has been collected.
The dataset is publically available on a website called Kaggle.com.
This dataset has been studied and analyzed using Logistic Regression machine learning algorithm.

# Description of Data



- **Code Link:**

  https://github.com/Chamindu77/TITANIC-Machine_Learning_from_Disaster/blob/main/titanic_ptoject.ipynb

- **Dataset:** Titanic - Machine Learning from Disaster

- **No. Data Points :** 891

- **No. Rows & Columns:**

```
titanic_df.shape
```

```
(891, 12)
```

# Description of Data

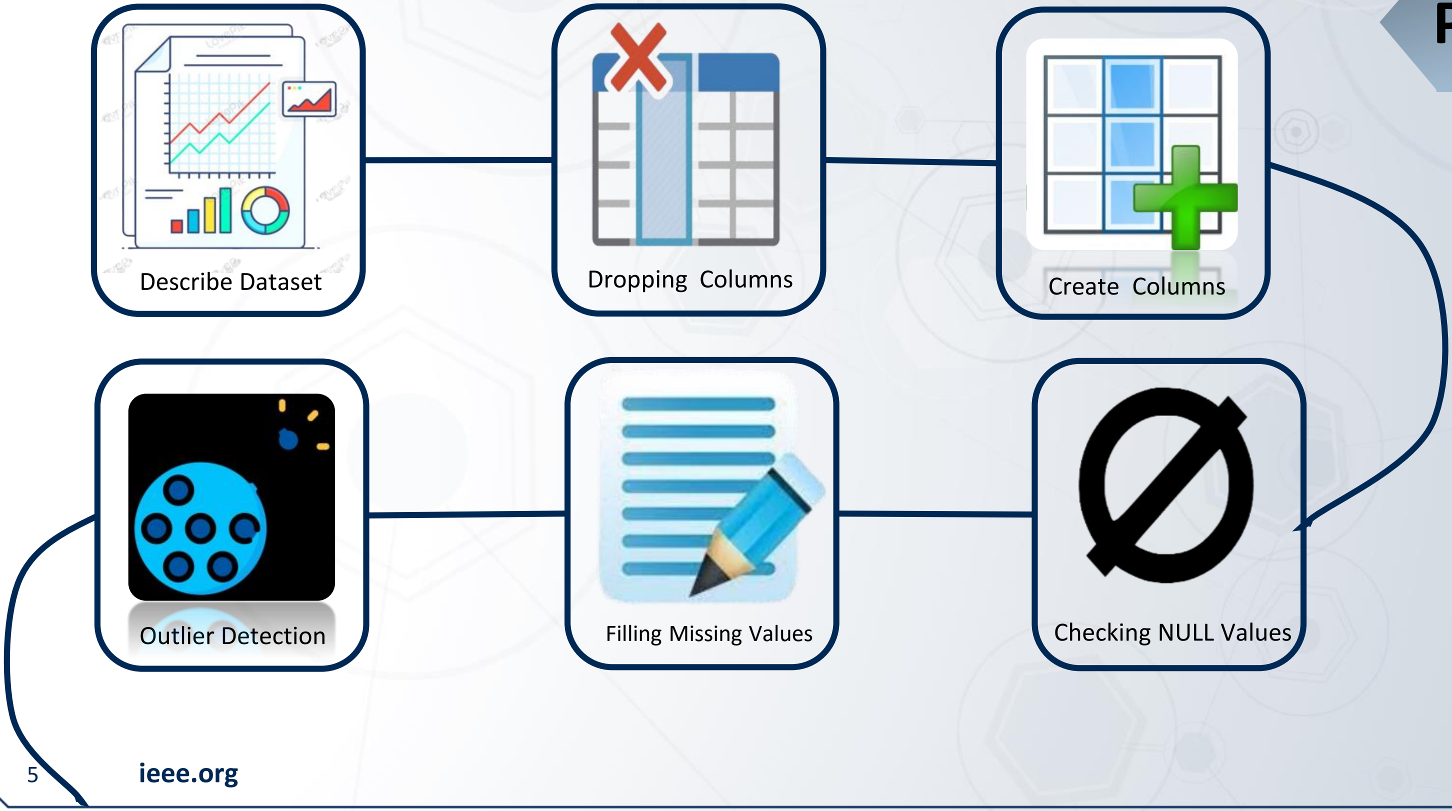


- **Data Dictionary :**

| **Variable** | **Definition** | **Key** |
|---|---|---|
| survival | Survival | 0 = No, 1 = Yes |
| pclass | Ticket class | 1 = 1st, 2 = 2nd, 3 = 3rd |
| sex | Sex | |
| Age | Age in years | |
| sibsp | # of siblings / spouses aboard the Titanic | |
| parch | # of parents / children aboard the Titanic | |
| ticket | Ticket number | |
| fare | Passenger fare | |
| cabin | Cabin number | |
| embarked | Port of Embarkation | C = Cherbourg, Q = Queenstown, S = Southampton |

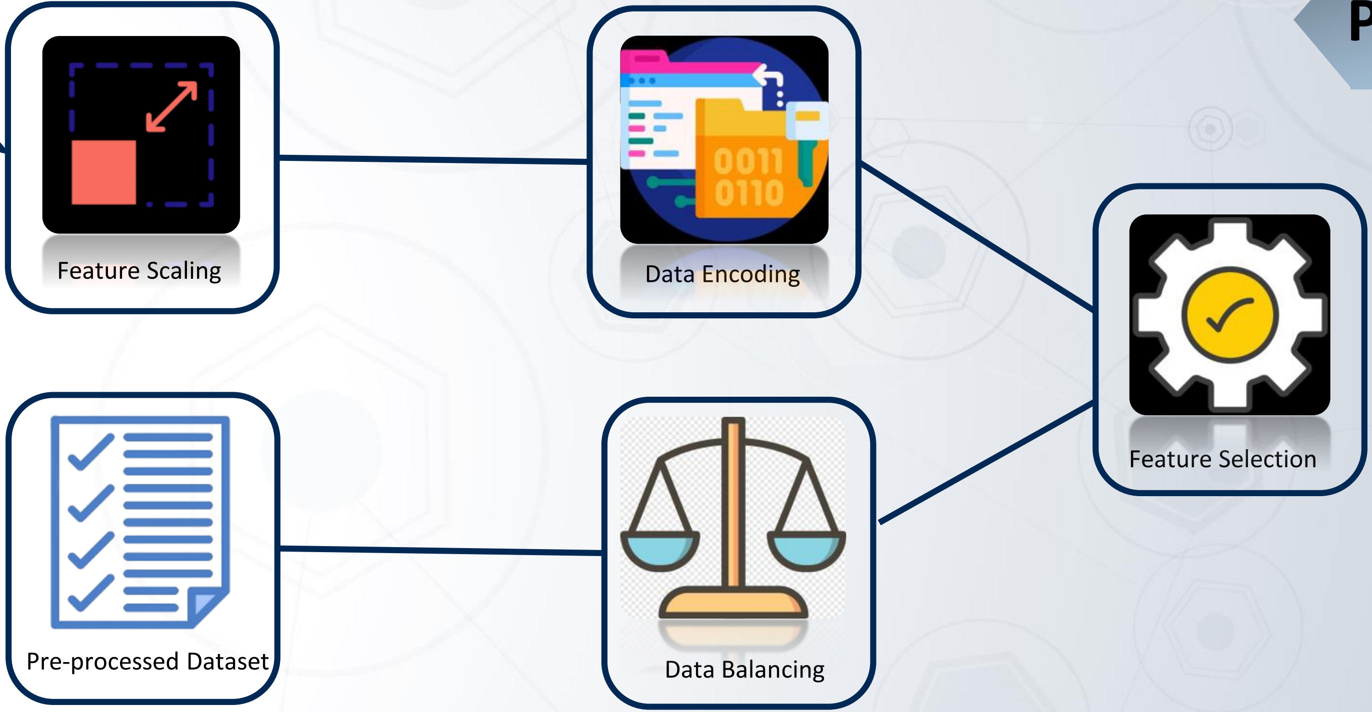
# Pre-Processing Data






IEEE
Advancing Technology for Humanity

# Pre-processing Data






IEEE
Advancing Technology for Humanity

Describe Dataset

# Describe Dataset



```python
titanic_df.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 |

IEEE
Advancing Technology for Humanity


# Describe Dataset



```python
titanic_df.describe()
```

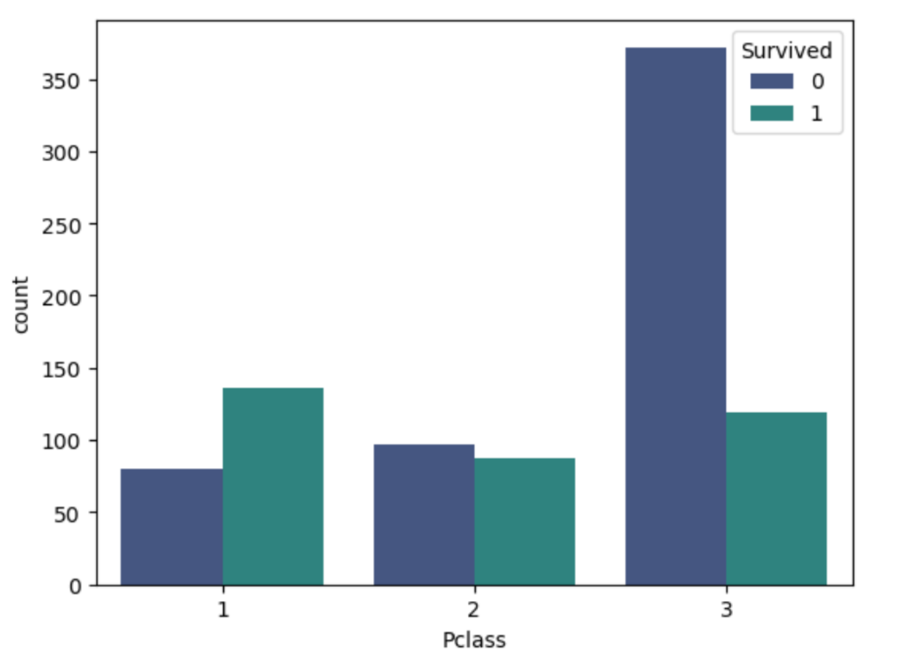
| | PassengerId | Survived | Pclass | Age | SibSp | Parch | Fare |
|---|---|---|---|---|---|---|---|
| **count** | 891.000000 | 891.000000 | 891.000000 | 714.000000 | 891.000000 | 891.000000 | 891.000000 |
| **mean** | 446.000000 | 0.383838 | 2.308642 | 29.699118 | 0.523008 | 0.381594 | 32.204208 |
| **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 |

IEEE Advancing Technology for Humanity

# Survival vs Ticket class

```python
custom_palette = sns.color_palette("viridis", 3)
sns.countplot(x='Pclass', hue='Survived', data=titanic_df_3, palette=custom_palette)
```

```python
# Group DataFrame by 'Pclass' and calculate mean survival rate for each class
titanic_df_3.groupby(['Pclass'], as_index=False)['Survived'].mean()
```




| Pclass | Survived |
|---|---|
| 1 | 0.629630 |
| 2 | 0.472826 |
| 3 | 0.242363 |

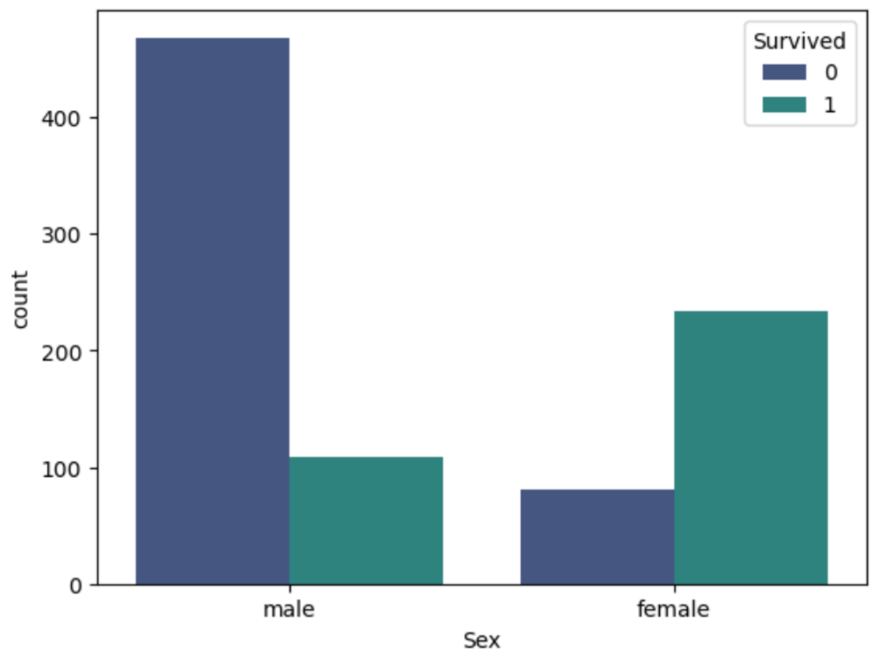


Advancing Technology for Humanity

# Survival vs Gender

```
custom_palette = sns.color_palette("viridis", 3)
sns.countplot(x='Sex', hue='Survived', data=titanic_df_6, palette=custom_palette)
```

```
# Group the DataFrame by the 'Sex' column and calculate the mean survival rate for gender
titanic_df_6.groupby(['Sex'], as_index=False)['Survived'].mean()
```




| Sex | Survived |
|---|---|
| female | 0.742038 |
| male | 0.188908 |

INTRODUCTION

**Pre-Processing**

Model Training

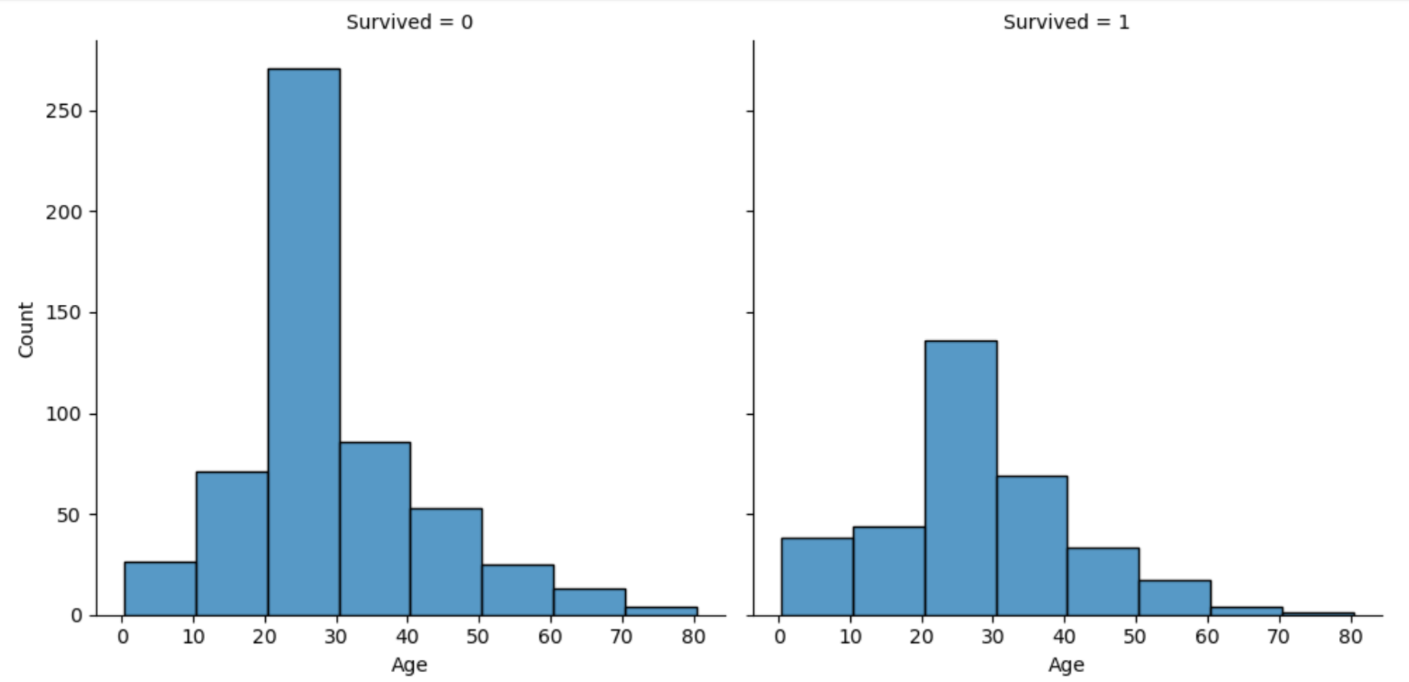
RESULTS

CONCLUSION

IEEE
Advancing Technology for Humanity

# Survival vs Age in years

```
sns.displot(titanic_df_6, x='Age', col='Survived', binwidth=10, height=5)
```




INTRODUCTION

**Pre-Processing**

Model Training

RESULTS

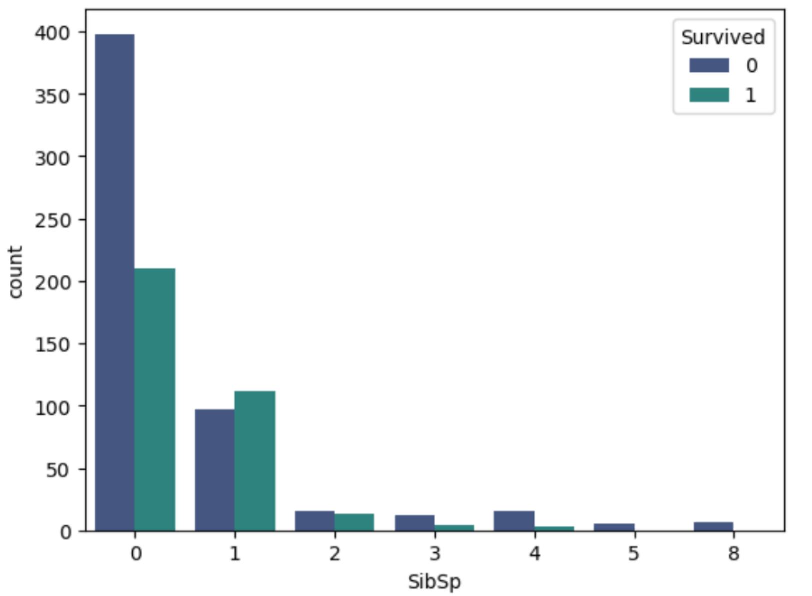
CONCLUSION

IEEE
Advancing Technology for Humanity

# Survival vs siblings

```python
custom_palette = sns.color_palette("viridis", 3)
sns.countplot(x='SibSp', hue='Survived', data=titanic_df_6, palette=custom_palette)
```

```python
# Group the DataFrame by the 'SibSp' column and calculate the mean survival rate for each unique value
titanic_df_6.groupby(['SibSp'], as_index=False)['Survived'].mean()
```




| SibSp | Survived |
|---|---|
| 0 | 0.345395 |
| 1 | 0.535885 |
| 2 | 0.464286 |
| 3 | 0.250000 |
| 4 | 0.166667 |
| 5 | 0.000000 |
| 8 | 0.000000 |

INTRODUCTION

Pre-Processing

Model Training

RESULTS

CONCLUSION

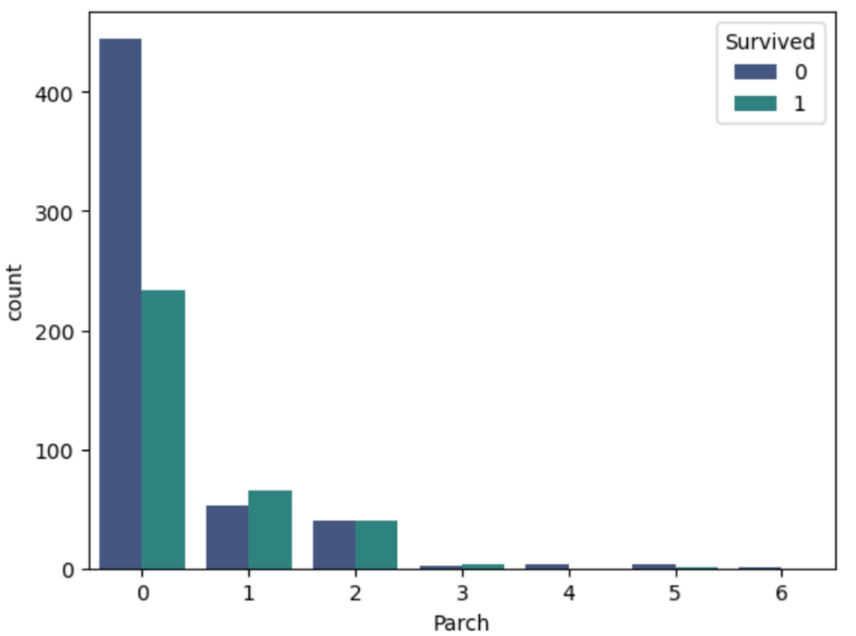
Advancing Technology for Humanity

# Survival vs Parents



```python
custom_palette = sns.color_palette("viridis", 3)
sns.countplot(x='Parch', hue='Survived', data=titanic_df_6, palette=custom_palette)
```

```python
# Group the DataFrame by the 'Parch' column and calculate the mean survival rate for each unique value
titanic_df_6.groupby(['Parch'], as_index=False)['Survived'].mean()
```



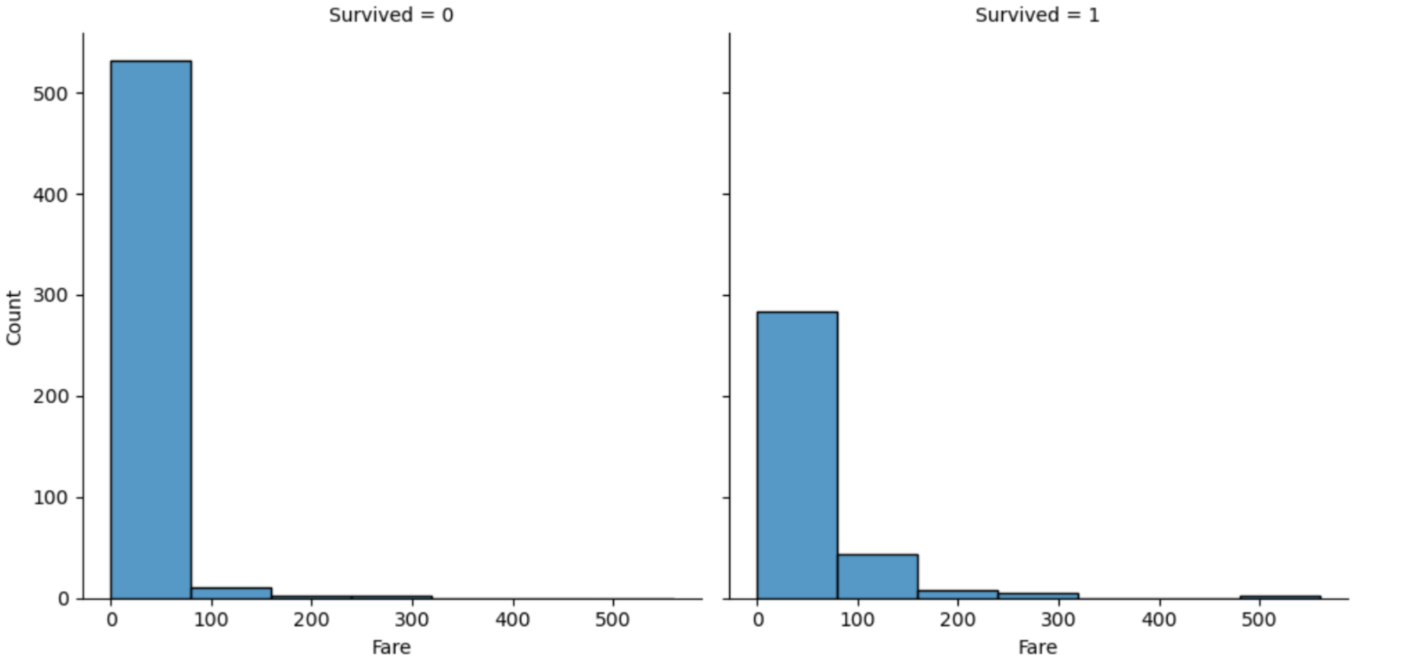

| | Parch | Survived |
|---|---|---|
| 0 | 0 | 0.343658 |
| 1 | 1 | 0.550847 |
| 2 | 2 | 0.500000 |
| 3 | 3 | 0.600000 |
| 4 | 4 | 0.000000 |
| 5 | 5 | 0.200000 |
| 6 | 6 | 0.000000 |

IEEE
Advancing Technology for Humanity

# Survival vs Passenger fare



```
sns.displot(titanic_df_9, x='Fare', col='Survived', binwidth=80, height=5)
```

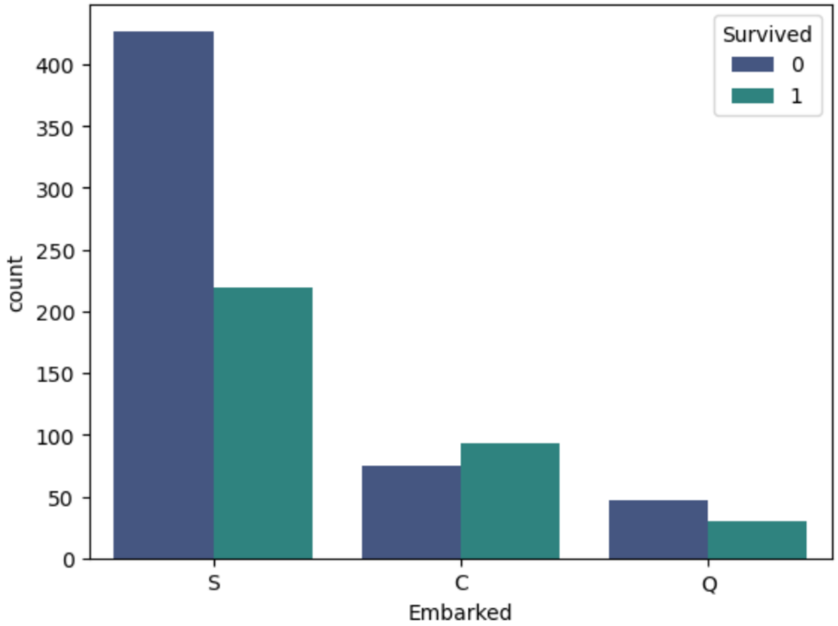


# Survival vs Port of Embarkation



```
# Calculate the mean survival rate for passengers grouped by their port of embarkation
titanic_df_10.groupby(['Embarked'], as_index=False)['Survived'].mean()
```

```
custom_palette = sns.color_palette("viridis", 3)
sns.countplot(x='Embarked', hue='Survived', data=titanic_df_10, palette=custom_palette)
```




| Embarked | Survived |
|---|---|
| C | 0.553571 |
| Q | 0.389610 |
| S | 0.339009 |

# Dropping Columns

Dropping Columns

Pre-Processing

INTRODUCTION

Model Training

RESULTS

CONCLUSION

- PassengerId Column

```
# Drop the 'PassengerId' column from the DataFrame 'titanic_df_3'
titanic_df_3 = titanic_df_3.drop(columns = 'PassengerId',axis=1)
```

- Ticket Column

```
# drop Ticket columns
titanic_df_9 = titanic_df_9.drop(columns = 'Ticket',axis=1)
```

- Cabin Column

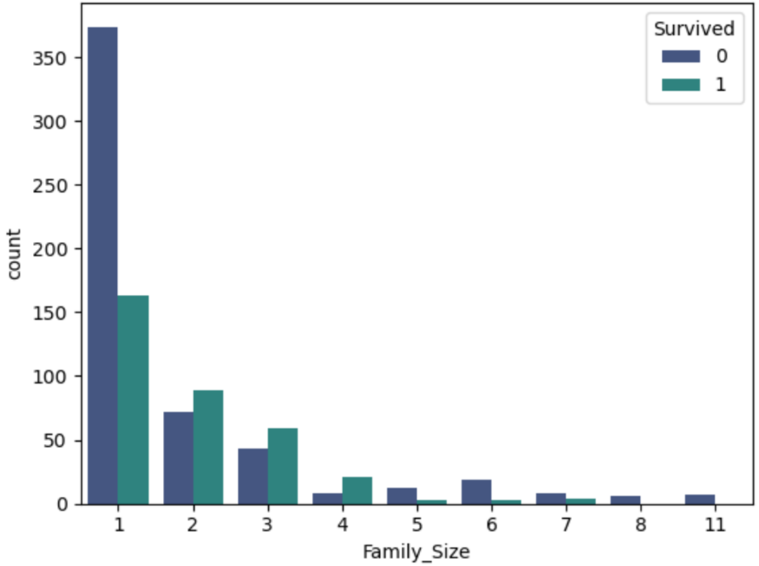
```
# drop Cabin columns
titanic_df_10 = titanic_df_10.drop(columns = 'Cabin',axis=1)
```

Advancing Technology for Humanity

# Create Columns – Family_Size

```python
# create family size column using SibSp column and Parch column
titanic_df_7['Family_Size'] = titanic_df_7['SibSp'] + titanic_df_7['Parch'] + 1
```

```python
# Calculate the mean survival rate for passengers grouped by their family size
titanic_df_7.groupby(['Family_Size'], as_index=False)['Survived'].mean()
```




| Family_Size | Survived |
|---|---|
| 1 | 0.303538 |
| 2 | 0.552795 |
| 3 | 0.578431 |
| 4 | 0.724138 |
| 5 | 0.200000 |
| 6 | 0.136364 |
| 7 | 0.333333 |
| 8 | 0.000000 |
| 11 | 0.000000 |

# Create Columns – Title

```
titanic_df_3['Name']

0                                Braund, Mr. Owen Harris
1      Cumings, Mrs. John Bradley (Florence Briggs Th...
2                                 Heikkinen, Miss. Laina
3           Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                               Allen, Mr. William Henry
                             ...
886                                Montvila, Rev. Juozas
887                         Graham, Miss. Margaret Edith
888             Johnston, Miss. Catherine Helen "Carrie"
889                                Behr, Mr. Karl Howell
890                                  Dooley, Mr. Patrick
Name: Name, Length: 891, dtype: object
```

```
titanic_df_4['Title']

0        Mr
1       Mrs
2      Miss
3       Mrs
4        Mr
       ...
886     Rev
887    Miss
888    Miss
889      Mr
890      Mr
Name: Title, Length: 891, dtype: object
```



IEEE
Advancing Technology for Humanity

# Create Columns – Title

| | Title |
|---|---|
| 0 | Capt |
| 1 | Col |
| 2 | Don |
| 3 | Dr |
| 4 | Jonkheer |
| 5 | Lady |
| 6 | Major |
| 7 | Master |
| 8 | Miss |
| 9 | Mlle |
| 10 | Mme |
| 11 | Mr |
| 12 | Mrs |
| 13 | Ms |
| 14 | Rev |
| 15 | Sir |
| 16 | the Countess |

```
# Map titles to corresponding categories:

#military - Capt, Col, Major
#noble - Jonkheer, the Countess, Don, Lady, Sir
#unmaried Female - MLle, Ms, Mme

# Replace titles with corresponding categories
titanic_df_5 = titanic_df_4.copy()
titanic_df_5['Title'] = titanic_df_5['Title'].replace({
    'Capt': 'Military',
    'Col': 'Military',
    'Major': 'Military',
    'Jonkheer': 'Noble',
    'the Countess': 'Noble',
    'Don': 'Noble',
    'Lady': 'Noble',
    'Sir': 'Noble',
    'Mlle': 'Noble',
    'Ms': 'Noble',
    'Mme': 'Noble'
})
```

| | Title | count |
|---|---|---|
| 0 | Dr | 7 |
| 1 | Master | 40 |
| 2 | Military | 5 |
| 3 | Miss | 182 |
| 4 | Mr | 517 |
| 5 | Mrs | 125 |
| 6 | Noble | 9 |
| 7 | Rev | 6 |

# Create Columns – Title

| | Title |
|---|---|
| 0 | Capt |
| 1 | Col |
| 2 | Don |
| 3 | Dr |
| 4 | Jonkheer |
| 5 | Lady |
| 6 | Major |
| 7 | Master |
| 8 | Miss |
| 9 | Mlle |
| 10 | Mme |
| 11 | Mr |
| 12 | Mrs |
| 13 | Ms |
| 14 | Rev |
| 15 | Sir |
| 16 | the Countess |

```python
# Map titles to corresponding categories:

#military - Capt, Col, Major
#noble - Jonkheer, the Countess, Don, Lady, Sir
#unmaried Female - MLle, Ms, Mme

# Replace titles with corresponding categories
titanic_df_5 = titanic_df_4.copy()
titanic_df_5['Title'] = titanic_df_5['Title'].replace({
    'Capt': 'Military',
    'Col': 'Military',
    'Major': 'Military',
    'Jonkheer': 'Noble',
    'the Countess': 'Noble',
    'Don': 'Noble',
    'Lady': 'Noble',
    'Sir': 'Noble',
    'Mlle': 'Noble',
    'Ms': 'Noble',
    'Mme': 'Noble'
})
```

| | Title | count |
|---|---|---|
| 0 | Dr | 7 |
| 1 | Master | 40 |
| 2 | Military | 5 |
| 3 | Miss | 182 |
| 4 | Mr | 517 |
| 5 | Mrs | 125 |
| 6 | Noble | 9 |
| 7 | Rev | 6 |

# Checking NULL Values



- ❑ Age column — Numerical Values → Mean
- Age column — Numerical Values → Median

- ❑ Cabin → Drop

- ❑ Embarked column — Categorical Values → Mode

```
titanic_df.isnull().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
```

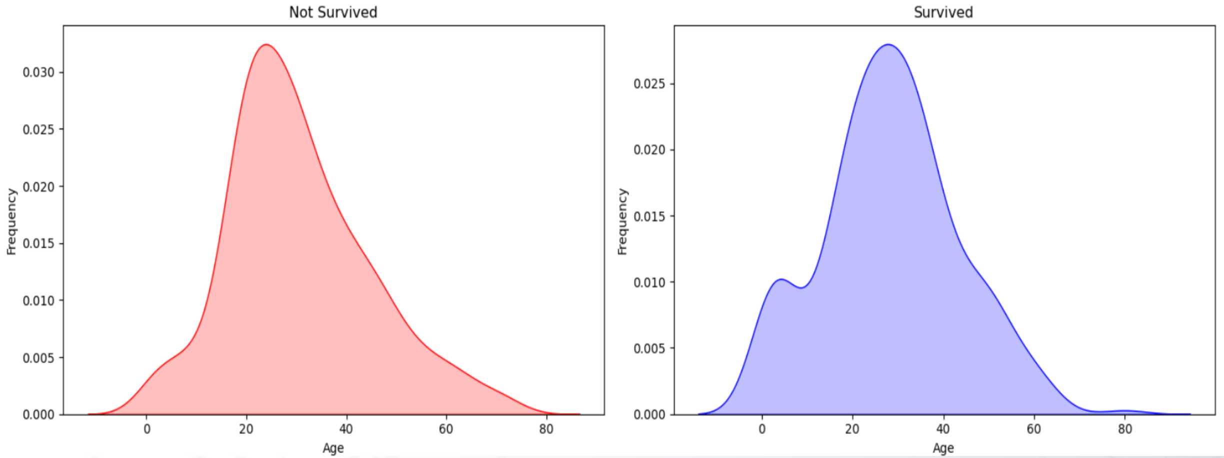
IEEE Advancing Technology for Humanity

# Filling Missing Values



- Age column

```python
# Fill missing values in the 'Age' column with the mean age of all passengers
titanic_df_1['Age'].fillna(titanic_df_1['Age'].mean(), inplace=True)
```




- Embarkation column

```python
# Fill missing values in the 'Embarked' column with the mode of the column
titanic_df_2['Embarked'].fillna(titanic_df_2['Embarked'].mode()[0], inplace=True)
```

IEEE
Advancing Technology for Humanity

# Outlier Detection

- ❑ Age Column
- ❑ Fare Column
- ❑ Family_Size Column

$$z = \frac{x - \mu}{\sigma}$$

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$$

$$\mu = \frac{\sum x}{N}$$

INTRODUCTION

**Pre-Processing**

Model Training

RESULTS

CONCLUSION

Advancing Technology for Humanity

# Outlier Detection

INTRODUCTION

**Pre-Processing**

Model Training

RESULTS

CONCLUSION

## ❑ Age Column

### Step - 01

```
age_mean = np.mean(titanic_df_10['Age'])
age_mean
```

```
29.69911764705882
```

```
age_std = np.std(titanic_df_10['Age'])
age_std
```

```
12.994716872789033
```

### Step - 02

```
titanic_df_11 = titanic_df_10.copy()
titanic_df_11['Age_z_score'] = (titanic_df_11['Age'] - age_mean)/age_std
```

### Step - 03

```
titanic_df_11['Age_z_score'].min()
```

```
-2.2531554887793948
```

```
titanic_df_11['Age_z_score'].max()
```

```
3.8708717431367314
```

# Outlier Detection



### Step - 04

```
age_outlier_indexes = []
age_outlier_indexes.extend(titanic_df_11.index[titanic_df_11['Age_z_score']>3].tolist())
age_outlier_indexes
```

```
[96, 116, 493, 630, 672, 745, 851]
```

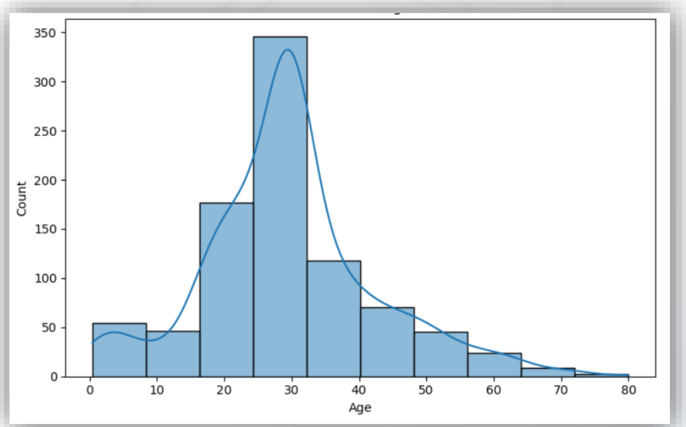
**Threshold Value**

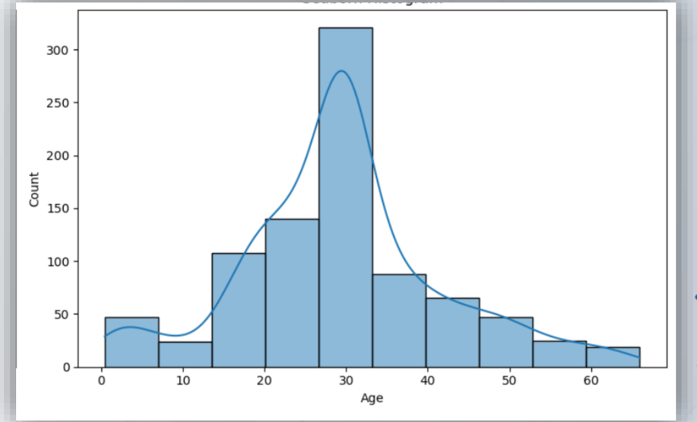
### Step - 05

```
titanic_df_11_new = titanic_df_11.drop(titanic_df_11.index[age_outlier_indexes])
```

### Step - 06

```
titanic_df_12 = titanic_df_11_new.drop('Age_z_score', axis=1)
```



# Feature Scaling

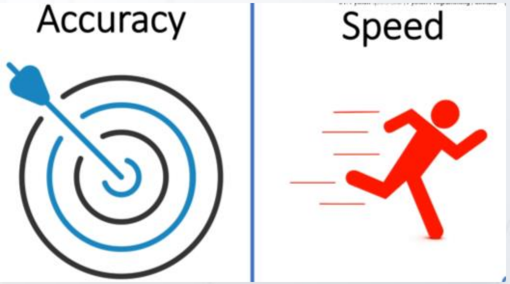


Normalization

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Standardization

$$x' = \frac{x - \mu}{\sigma}$$



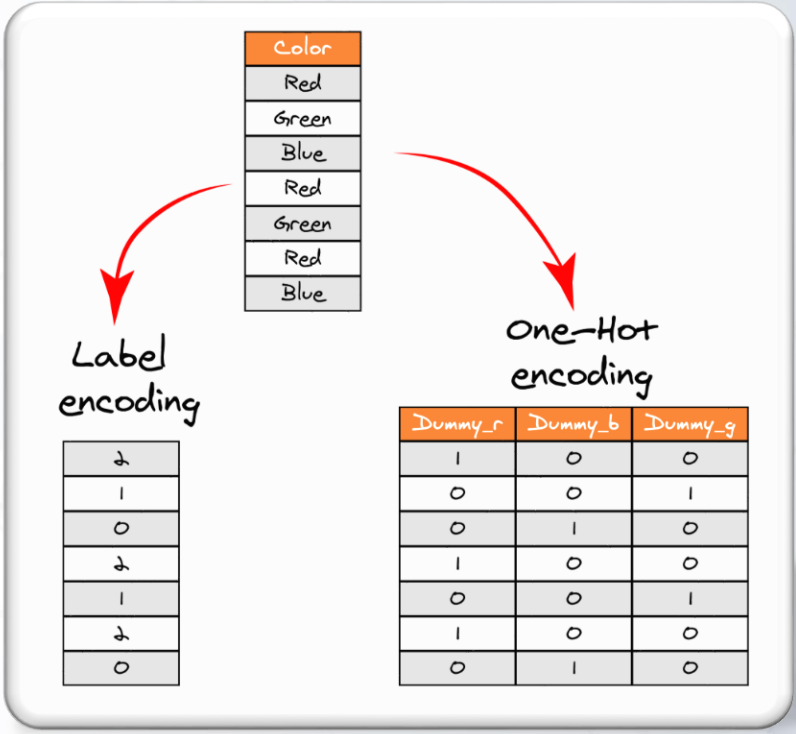
# Feature Scaling

- ❑ Age Column
- ❑ Fare Column

```
titanic_df_17 =titanic_df_16.copy()

scaler = MinMaxScaler(feature_range=(0, 10))

numeric_columns = ['Age', 'Fare']

for column in numeric_columns:
    column_data = titanic_df_17[column].values.reshape(-1, 1)
    scaled_data = scaler.fit_transform(column_data)
    titanic_df_17[column] = scaled_data
```

# Data Encoding

- Label Encoding
- One Hot Encoding






IEEE Advancing Technology for Humanity

# Data Encoding using One Hot Encoding

- ❑ Pclass Column
- ❑ Embarked Column
- ❑ Sex Column
- ❑ Title Column

```
titanic_df_18 = titanic_df_17.copy()

# Select categorical columns
categorical_columns = ['Pclass', 'Sex', 'Embarked', 'Title']

# Extract categorical features
categorical_data = titanic_df_18[categorical_columns]

# Initialize OneHotEncoder
one_hot_encoder = OneHotEncoder(sparse=False, drop='first')

# Fit and transform the categorical features
encoded_data = one_hot_encoder.fit_transform(categorical_data)

# Get the feature names from OneHotEncoder
feature_names = one_hot_encoder.get_feature_names_out(categorical_columns)

# Create DataFrame with encoded features
encoded_df = pd.DataFrame(encoded_data, columns=feature_names, index=titanic_df_18.index)

# Drop original categorical columns from titanic_df_18
titanic_df_18.drop(columns=categorical_columns, inplace=True)

# Concatenate titanic_df_18 with encoded_df
titanic_df_18 = pd.concat([titanic_df_18, encoded_df], axis=1)
```

# Feature Selection

- ❑ Correlation Matrix and Heat Map Visualization
- ❑ Mutual Information Feature Selection

INTRODUCTION

**Pre-Processing**

Model Training

RESULTS

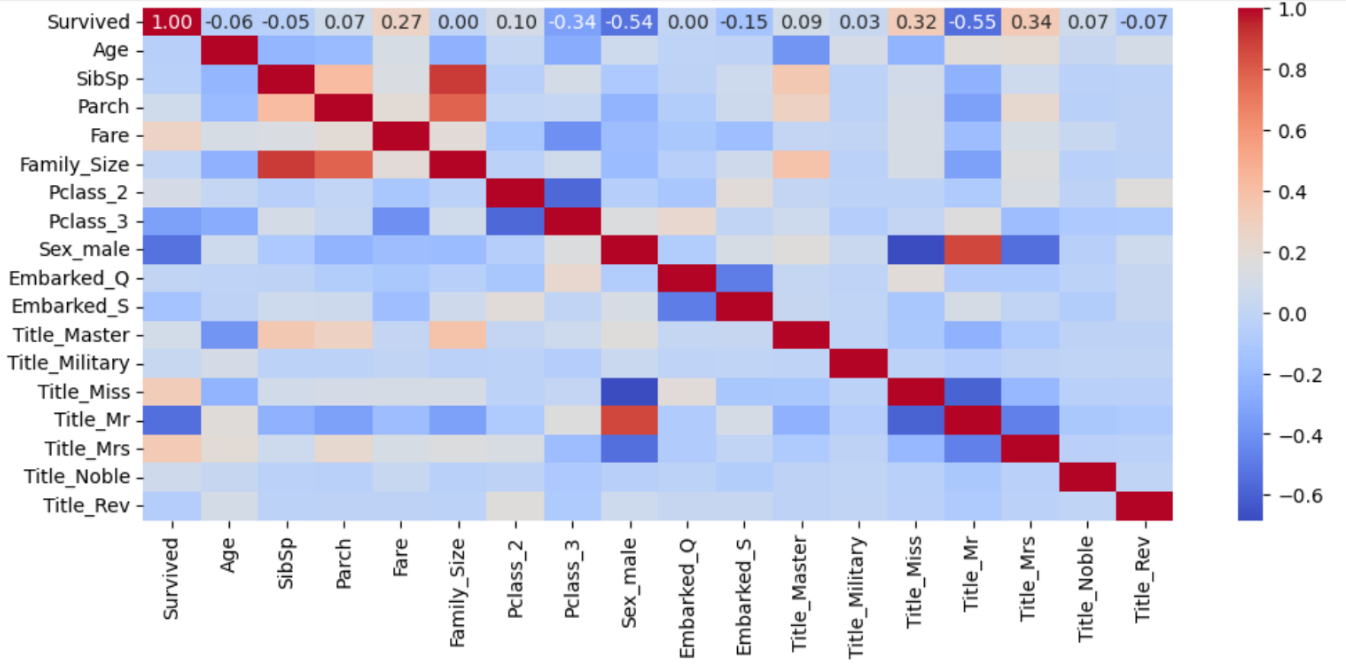
CONCLUSION

| Mutual Information | Correlation Matrix |
|---|---|
| • Captures both linear and non-linear relationships between variables<br>• Considers the direct relationship between features and the target variable<br>• Computationally more expensive compared to correlation matrix<br>• May require tuning parameters like the number of features to select (k) | • Helps identify highly correlated features<br>• Easy to understand and visualize<br>• Only captures linear relationships between variables<br>• May not identify non-linear relationships |

IEEE
Advancing Technology for Humanity

# Feature Selection

- ❑ **Correlation Matrix and Heat Map Visualization**




INTRODUCTION

**Pre-Processing**

Model Training

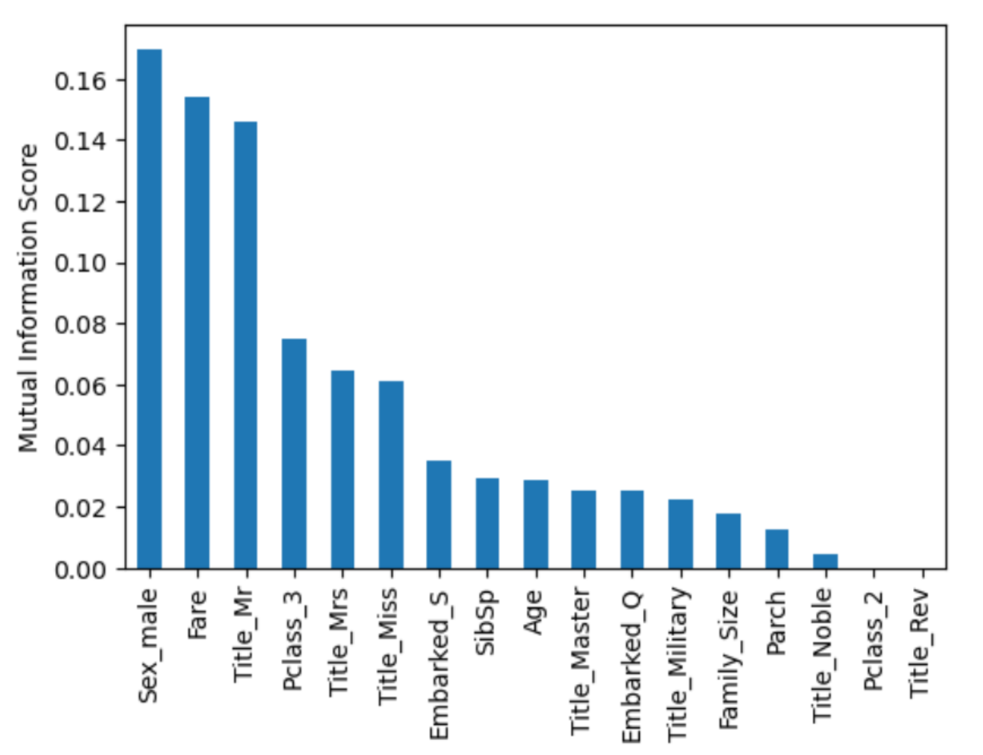
RESULTS

CONCLUSION

Advancing Technology for Humanity

# Feature Selection

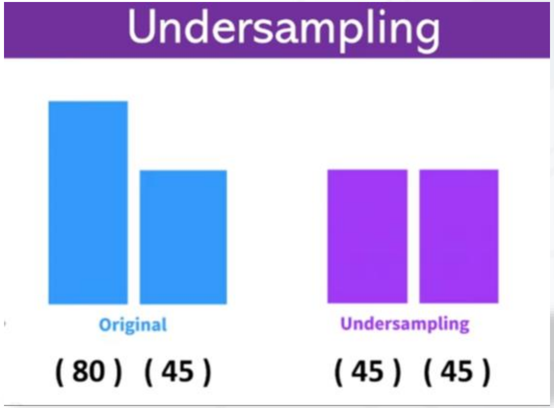
- Mutual Information Feature Selection

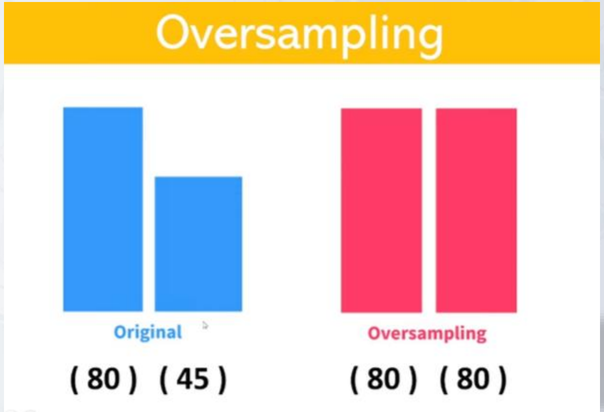


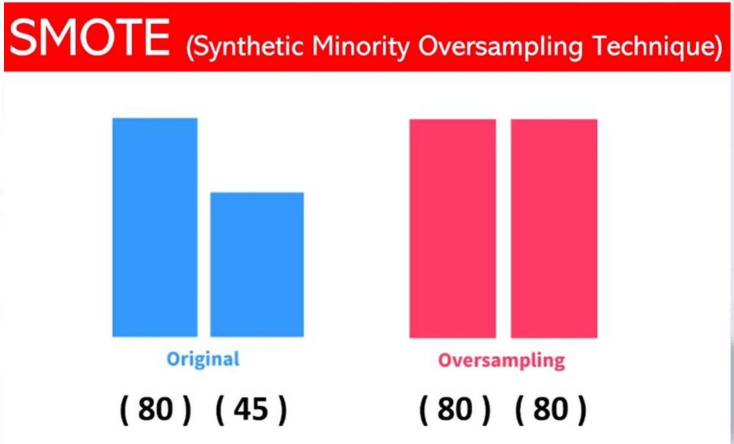



Advancing Technology for Humanity

# Data Balancing



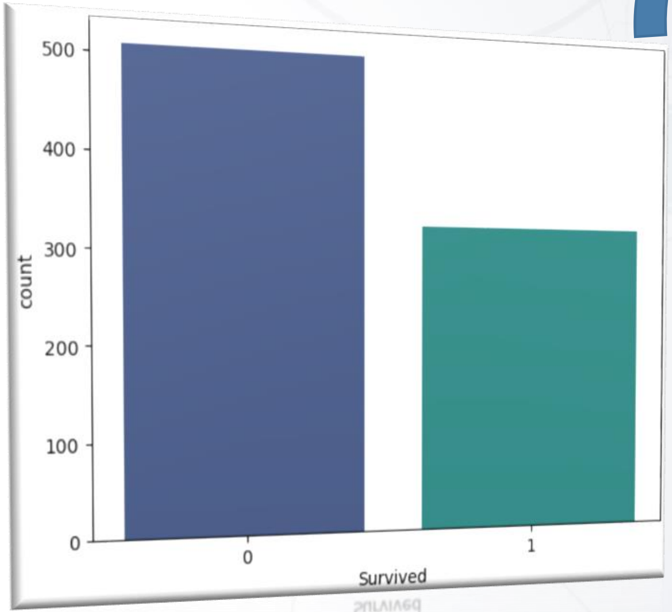


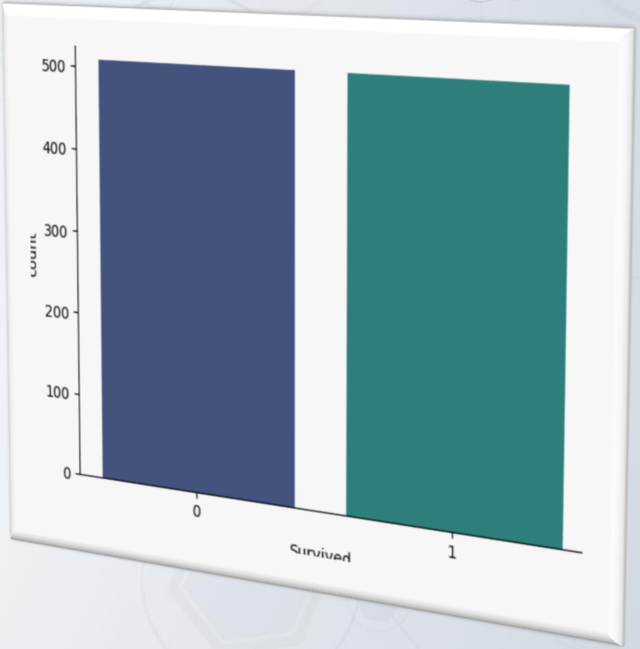

IEEE
Advancing Technology for Humanity

# Data Balancing

**SMOTE** (Synthetic Minority Oversampling Technique)



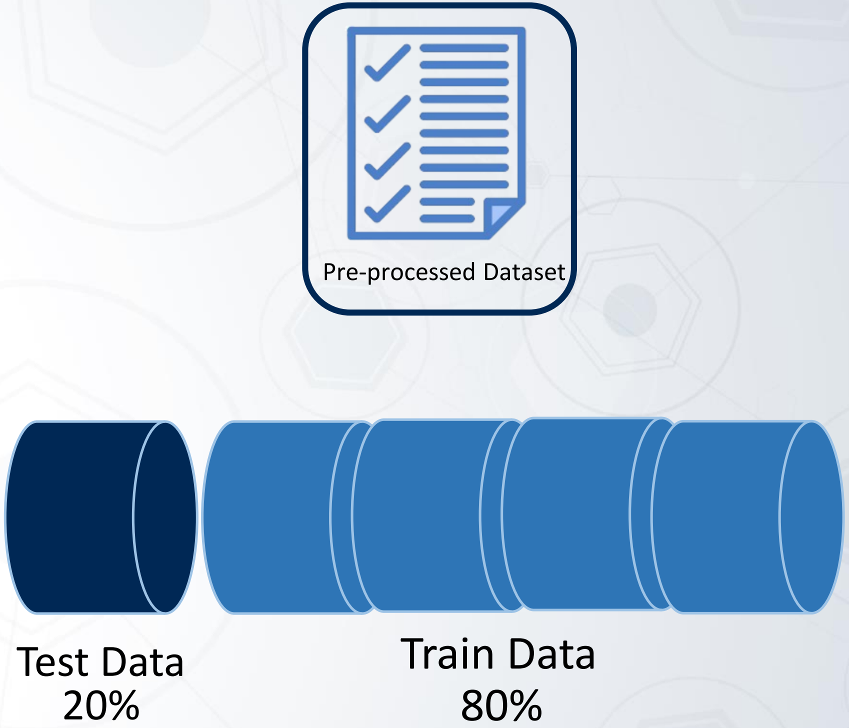



IEEE
Advancing Technology for Humanity

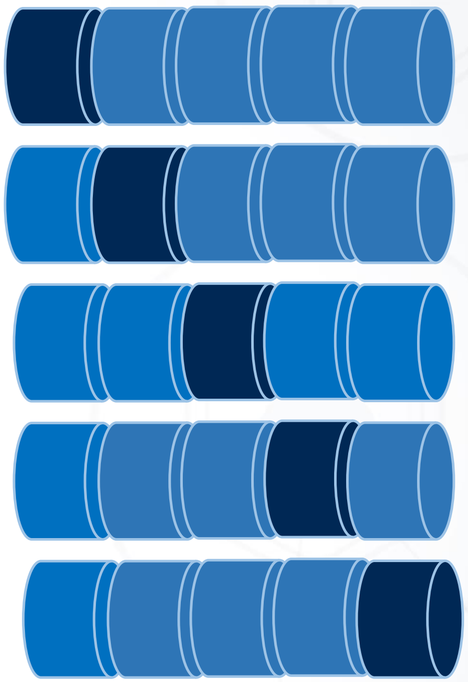
# Data Splitting





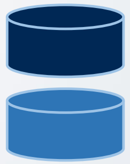
# Model Evaluation Technique





**K-Fold Cross-Validation**

Cross-validation is a technique used in machine learning to check how well a model can generalize to new, or unseen data.




IEEE
Advancing Technology for Humanity

# Model Evaluation Technique



- Add K-Fold Cross-Validation Technique for Better Accuracy

```
log_reg_model = LogisticRegression()
num_splits = 3

kf = KFold(n_splits=num_splits, shuffle=True, random_state=21)

cv_scores = cross_val_score(log_reg_model, X, y, cv=kf).mean()

print("Cross-validation scores:", cv_scores)
```

```
Cross-validation scores: 0.8128904249871992
```

# Modifications

- **Parameters:**

Kernal Function
Regularization Parameter
Tolerance to Convergence
Class Weight
Penalty

Variable

Solver
Multi-class Handling

Fixed



IEEE
Advancing Technology for Humanity

# Modifications



## ❑ Add Hyper Parameter Tuning

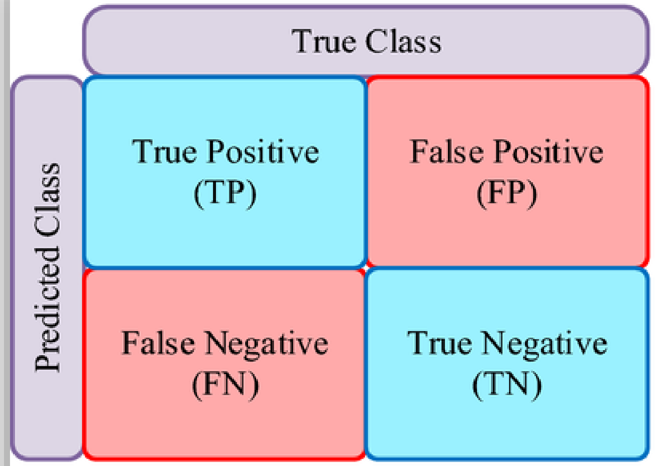
```
param_grid = {
        'penalty': ['l1', 'l2', 'elasticnet', 'none'],
        'C': np.logspace(-2, 2, 5),
        'solver': ['lbfgs', 'liblinear', 'saga'],
        'max_iter': [100, 500, 1000],
        'l1_ratio': np.linspace(0, 1, 3),
}

num_splits = 3
kf = KFold(n_splits=num_splits, shuffle=True, random_state=21)

grid_search = GridSearchCV(log_reg_model, param_grid, cv=kf, scoring='accuracy')
grid_search.fit(X_train, y_train)

best_params = grid_search.best_params_

cv_scores = grid_search.best_score_
best_model = grid_search.best_estimator_

print("Best Hyperparameters" )
best_params
```

```
Best Hyperparameters
{'C': 0.01,
 'l1_ratio': 0.0,
 'max_iter': 100,
 'penalty': 'none',
 'solver': 'lbfgs'}
```

# Performance Evaluation Metrics



- ❑ **Accuracy**
- ❑ **Precision**
- ❑ **Recall**
- ❑ **F1 - Score**




| Function Name | Formula |
|---|---|
| Accuracy | $\frac{TP+TN}{TP+TN+FP+FN}$ |
| Precision | $\frac{TP}{TP+FP}$ |
| Recall | $\frac{TP}{TP+FN}$ |
| F1 score | $\frac{2.Precision.Recall}{Precision+Recall}$ |

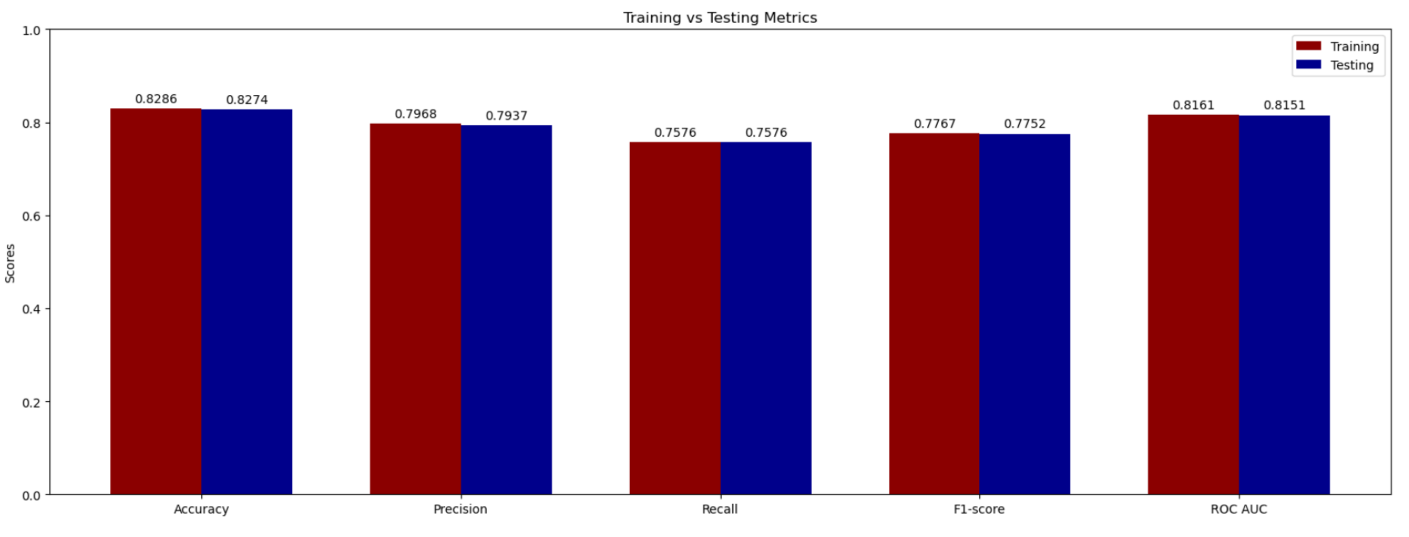
Advancing Technology for Humanity

# Key Results

- Before Hyper parameter Tuning

```
Training Accuracy: 0.8286
Training Precision: 0.7968
Training Recall: 0.7576
Training F1-score: 0.7767
Training ROC AUC Score: 0.8161
```

```
Testing Accuracy: 0.8274
Testing Precision: 0.7937
Testing Recall: 0.7576
Testing F1-score: 0.7752
Testing ROC AUC Score: 0.8151
```




INTRODUCTION

Pre-Processing

Model Training

**RESULTS**

CONCLUSION

IEEE Advancing Technology for Humanity

# Key Results



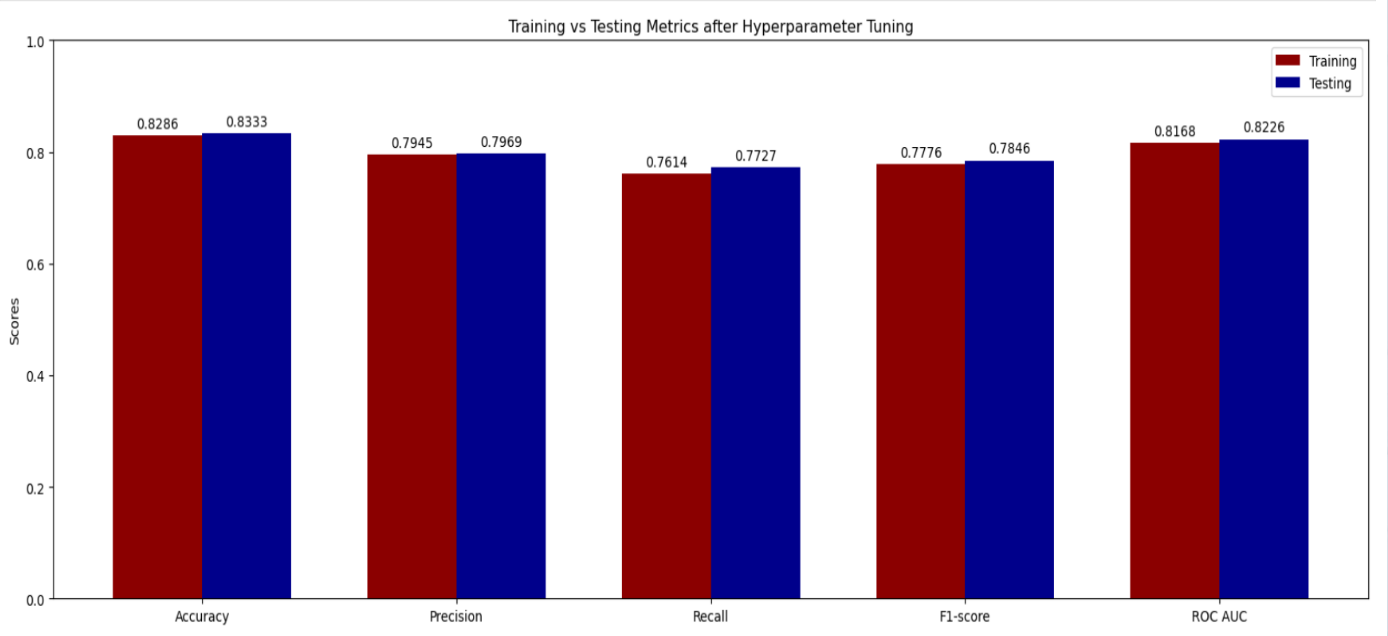
- After Hyper parameter Tuning

```
Training Accuracy after Hyperparameter Tuning: 0.8286
Training Precision after Hyperparameter Tuning: 0.7945
Training Recall after Hyperparameter Tuning: 0.7614
Training F1-score after Hyperparameter Tuning: 0.7776
Training ROC AUC Score after Hyperparameter Tuning: 0.8168
```

```
Testing Accuracy after Hyperparameter Tuning: 0.8333
Testing Precision after Hyperparameter Tuning: 0.7969
Testing Recall after Hyperparameter Tuning: 0.7727
Testing F1-score after Hyperparameter Tuning: 0.7846
Testing ROC AUC Score after Hyperparameter Tuning: 0.8226
```

# Key Results

- After Hyper parameter Tuning




INTRODUCTION

Pre-Processing

Model Training

RESULTS

CONCLUSION

IEEE Advancing Technology for Humanity

# Thank You