

```
In [3]: from IPython.display import Image, display
display(Image(filename='Project.2.jpg'))
```



```
In [ ]: Dataset link https://www.kaggle.com/datasets/shuofxz/titanic-machine-learning-from-
```

```
In [ ]: Load the dataset
```

```
In [6]: df.shape, df.isnull().sum().sort_values(ascending=False).head(20)
```

```
Out[6]: ((891, 12),
 Cabin      687
 Age        177
 Embarked    2
 PassengerId 0
 Name        0
 Pclass     0
 Survived    0
 Sex         0
 Parch       0
 SibSp       0
 Fare        0
 Ticket     0
 dtype: int64)
```

```
In [ ]: Handle missing value using median & mode
```

```
In [7]: df.isnull().sum()
```

```
Out[7]: 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
```

```
In [ ]: Clean the data
```

```
In [46]: import pandas as pd
import os, sys

print("Python:", sys.version)
print("Working directory:", os.getcwd())

csv_files = [f for f in os.listdir() if f.lower().endswith(".csv")]
print("CSV files found:", csv_files)

file_name = "Titanic Machine Learning from Disaster.csv"

df = pd.read_csv(file_name)
print("\n Loaded successfully!")
print("Type:", type(df))
print("Shape:", df.shape)

print("\nColumns (raw):")
for c in df.columns:
    print(repr(c))

df.columns = df.columns.astype(str).str.strip()

cols_to_drop = ["Cabin", "Ticket", "Name", "PassengerId"]
df = df.drop(columns=[c for c in cols_to_drop if c in df.columns])

if "Age" in df.columns:
    df["Age"] = df["Age"].fillna(df["Age"].median())
if "Embarked" in df.columns:
    df["Embarked"] = df["Embarked"].fillna(df["Embarked"].mode()[0])

print("\n Cleaning completed!")
print("Missing values (top):")
print(df.isnull().sum().sort_values(ascending=False).head(15))
```

```
except Exception as e:  
    print("\n ERROR happened:")  
    print(type(e).__name__, ":", e)
```

Python: 3.13.9 | packaged by Anaconda, Inc. | (main, Oct 21 2025, 19:09:58) [MSC v.1929 64 bit (AMD64)]

Working directory: C:\Users\Waleed.Abdelgawad

CSV files found: ['Customer_Review.csv', 'heart.csv', 'heart_cleaned.csv', 'Titanic Machine Learning from Disaster.csv']

✅ Loaded successfully!

Type: <class 'pandas.core.frame.DataFrame'>

Shape: (891, 12)

Columns (row):

'PassengerId'

'Survived'

'Pclass'

'Name'

'Sex'

'Age'

'SibSp'

'Parch'

'Ticket'

'Fare'

'Cabin'

'Embarked'

✅ Cleaning completed!

Missing values (top):

Survived 0

Pclass 0

Sex 0

Age 0

SibSp 0

Parch 0

Fare 0

Embarked 0

dtype: int64

In []: Apply ML Models with GridSearch

In [47]: **import** pandas **as** pd

```
df = pd.get_dummies(df, columns=["Sex", "Embarked"], drop_first=True)  
df.head()
```

Out[47]:

	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_male	Embarked_Q	Embarked_S
0	0	3	22.0	1	0	7.2500	True	False	True
1	1	1	38.0	1	0	71.2833	False	False	False
2	1	3	26.0	0	0	7.9250	False	False	True
3	1	1	35.0	1	0	53.1000	False	False	True
4	0	3	35.0	0	0	8.0500	True	False	True

In [48]:

```

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, f1_score, classification_report

lr = LogisticRegression(max_iter=2000)

param_grid_lr = {
    "C": [0.1, 1, 10]
}

grid_lr = GridSearchCV(lr, param_grid_lr, cv=5, scoring="f1")
grid_lr.fit(X_train, y_train)

pred_lr = grid_lr.predict(X_test)

print("Best Parameters:", grid_lr.best_params_)
print("Accuracy:", accuracy_score(y_test, pred_lr))
print("F1 Score:", f1_score(y_test, pred_lr))
print(classification_report(y_test, pred_lr))

```

Best Parameters: {'C': 0.1}

Accuracy: 0.7988826815642458

F1 Score: 0.7142857142857143

	precision	recall	f1-score	support
0	0.80	0.89	0.84	110
1	0.79	0.65	0.71	69
accuracy			0.80	179
macro avg	0.80	0.77	0.78	179
weighted avg	0.80	0.80	0.79	179

In []: Feature Selection

In []: Select KBest feature

In [51]:

```

from sklearn.feature_selection import SelectKBest, chi2

skb = SelectKBest(score_func=chi2, k=5)
X_train_skb = skb.fit_transform(X_train, y_train)
X_test_skb = skb.transform(X_test)

```

```
selected_features_skb = X_train.columns[skb.get_support()]
selected_features_skb
```

Out[51]: Index(['Pclass', 'SibSp', 'Parch', 'Fare', 'Sex_male'], dtype='object')

In []: Select RFE

```
In [52]: from sklearn.feature_selection import RFE

rfe = RFE(estimator=LogisticRegression(max_iter=2000), n_features_to_select=5)
X_train_rfe = rfe.fit_transform(X_train, y_train)
X_test_rfe = rfe.transform(X_test)

selected_features_rfe = X_train.columns[rfe.get_support()]
selected_features_rfe
```

Out[52]: Index(['Pclass', 'SibSp', 'Parch', 'Fare', 'Sex_male'], dtype='object')

In []: Apply ML Models

```
In [53]: lr.fit(X_train_skb, y_train)
pred_skb = lr.predict(X_test_skb)

print("SelectKBest Accuracy:", accuracy_score(y_test, pred_skb))
print("SelectKBest F1:", f1_score(y_test, pred_skb))
```

SelectKBest Accuracy: 0.8100558659217877

SelectKBest F1: 0.7301587301587301

```
In [54]: lr.fit(X_train_rfe, y_train)
pred_rfe = lr.predict(X_test_rfe)

print("RFE Accuracy:", accuracy_score(y_test, pred_rfe))
print("RFE F1:", f1_score(y_test, pred_rfe))
```

RFE Accuracy: 0.8100558659217877

RFE F1: 0.7301587301587301