In [1]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, recall_score, classification_report
import warnings
warnings.filterwarnings("ignore", category=UserWarning, module="sklearn")
```

In [2]:

```
# Load the dataset

df = pd.read_csv("/kaggle/input/titanic-survival-dataset/Titanic-Dataset.csv")
```

In [3]:

```
df.head()
```

Out[3]:

	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

In [4]:

```
# Select features and target
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']
target = 'Survived'
X = df[features]
y = df[target]
#Preprocessing & train/test
numeric_features = ['Age', 'SibSp', 'Parch', 'Fare']
categorical_features = ['Sex', 'Embarked']

numeric_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="median")),
    ("scaler", StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("encoder", OneHotEncoder(drop="first", handle_unknown="ignore"))
])

preprocessor = ColumnTransformer(
    transformers=[
        ("num", numeric_transformer, numeric_features),
        ("cat", categorical_transformer, categorical_features)
    ]
)
```

In [5]:

```
df.head()
```

Out[5]:

	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

In [6]:

```
# Logistic regression model
model = Pipeline(steps=[
    ("preprocessor", preprocessor),
    ("classifier", LogisticRegression(max_iter=100, solver="liblinear"))
])

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
```

In [7]:

```
#Train & Evaluate model
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("=== Task 2: Performance ===")
print("Accuracy:", accuracy)
```

In [7]:

```
#Train & Evaluate model
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("=== Task 2: Performance ===")
print("Accuracy:", accuracy)
print("Recall:", recall)
print("Classification Report:\n", classification_report(y_test, y_pred))


#Extract coefficients

log_reg = model.named_steps['classifier']
ohe = model.named_steps['preprocessor'].named_transformers_['cat'].named_steps['encoder']

feature_names = numeric_features + list(ohe.get_feature_names_out(categorical_features))
theta = pd.Series(
    np.append(log_reg.intercept_, log_reg.coef_[0]),
    index=['intercept'] + feature_names
)

print("\n=== Task 3: Theta coefficients ===")
print(theta)
```

```
print(theta)
```

```
=== Task 2: Performance ===
Accuracy: 0.7988826815642458
Recall: 0.6666666666666666
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.88      0.84       110
           1       0.78      0.67      0.72        69

    accuracy                           0.80       179
   macro avg       0.79      0.77      0.78       179
weighted avg       0.80      0.80      0.80       179


=== Task 3: Theta coefficients ===
intercept     1.332172
Age          -0.282279
SibSp        -0.335404
Parch        -0.171092
Fare          0.746873
Sex_male     -2.447513
Embarked_Q   -0.224984
Embarked_S   -0.410692
dtype: float64
```

In [8]:

```
#Prediction on new records
new_data = pd.DataFrame([
    {'Pclass': 1, 'Sex': 'female', 'Age': 29, 'SibSp': 0, 'Parch': 0, 'Fare': 72.0, 'Embarked': 'C'},
    {'Pclass': 3, 'Sex': 'male', 'Age': 22, 'SibSp': 1, 'Parch': 0, 'Fare': 7.25, 'Embarked': 'S'},
    {'Pclass': 2, 'Sex': 'male', 'Age': 45, 'SibSp': 0, 'Parch': 0, 'Fare': 13.0, 'Embarked': 'S'}
])

preds = model.predict(new_data)
new_data['Prediction'] = np.where(preds == 1, "survived", "not survived")
print(new_data)
```

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Prediction
0	1	female	29	0	0	72.00	C	survived
1	3	male	22	1	0	7.25	S	not survived
2	2	male	45	0	0	13.00	S	not survived

In [9]:

```
#Variations (Task 5)
configs = [
    {"test_size": 0.30, "max_iter": 100},
    {"test_size": 0.20, "max_iter": 50},
    {"test_size": 0.20, "max_iter": 500},
    {"test_size": 0.40, "max_iter": 200},
]

print("\n=== Task 5: Variations ===")
for cfg in configs:
    model_var = Pipeline(steps=[
        ("preprocessor", preprocessor),
        ("classifier", LogisticRegression(max_iter=cfg["max_iter"], solver="liblinear"))
    ])

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=cfg["test_size"], random_state=42, stratify=y
    )

    model_var.fit(X_train, y_train)
    y_pred = model_var.predict(X_test)

    acc = accuracy_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)

    print(f"Config test_size={cfg['test_size']}, max_iter={cfg['max_iter']} -> Accuracy={acc:.4f}, Recall
={rec:.4f}")
```

```
=== Task 5: Variations ===
Config test_size=0.3, max_iter=100 -> Accuracy=0.7910, Recall=0.6893
Config test_size=0.2, max_iter=50 -> Accuracy=0.7989, Recall=0.6667
Config test_size=0.2, max_iter=500 -> Accuracy=0.7989, Recall=0.6667
Config test_size=0.4, max_iter=200 -> Accuracy=0.8039, Recall=0.6861
```