In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.metrics import accuracy_score
from xgboost import XGBClassifier
from scipy.stats import uniform, randint
```

In [2]:
```
train = pd.read_csv("Data/train.csv")
test = pd.read_csv("Data/test.csv")
```

In [3]:
```
# data cleaning, feature eng

def preprocess_data(df, all_columns=None, is_training=True):
    df = df.copy()

    # handle missing values, create categorical bins, process features
    df = df.assign(
        Age=lambda x: x["Age"].fillna(x["Age"].median()),
        Fare=lambda x: x["Fare"].fillna(x["Fare"].median()),
        Embarked=lambda x: x["Embarked"].fillna(x["Embarked"].mode()[0]),
        Age_Category=lambda x: pd.qcut(x["Age"], q=5, labels=False, duplicates="drop"),
        Fare_Category=lambda x: pd.qcut(x["Fare"], q=5, labels=False, duplicates="drop"),
        Ticket_Prefix=lambda x: x["Ticket"].str.extract(r'([A-Za-z./]+)', expand=False).fillna("None"),
        Cabin_Letter=lambda x: x["Cabin"].str.extract(r'([A-Za-z])', expand=False).fillna("M"),
        Has_Cabin=lambda x: x["Cabin"].notna().astype(int)
        )

    for col in ["Age_Category", "Fare_Category", "Sex", "Pclass", "Embarked", "Ticket_Prefix", "Cabin_Letter"]:
        df = pd.get_dummies(df, columns=[col], drop_first=(col not in ["Ticket_Prefix", "Cabin_Letter"]))

    df = df.drop(columns=["Name", "Ticket", "Cabin"], errors="ignore")

    # match column structure for test data
    if all_columns is not None:
        df = df.reindex(columns=all_columns, fill_value=0)

    if is_training and "Survived" in df.columns:
        df = df.drop(columns=["Survived"])

    return df
```

In [4]:
```
# data splitting

train_features = preprocess_data(train, is_training=True)
all_columns = train_features.columns.tolist()
test_features = preprocess_data(test, all_columns=all_columns, is_training=False)

X = train_features
y = train["Survived"]
X_test = test_features

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
```

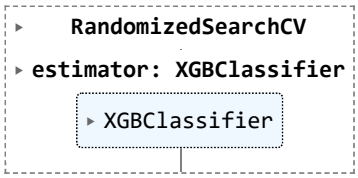
In [5]:
```
param_dist = {
    'n_estimators': randint(50, 500),
    'max_depth': randint(3, 15),
    'learning_rate': uniform(0.01, 0.3),
    'subsample': uniform(0.5, 0.5),
    'colsample_bytree': uniform(0.5, 0.5),
}
```

In [6]:
```
model = XGBClassifier(eval_metric="logloss", random_state=42)
```

In [7]:
```
random_search = RandomizedSearchCV(
    model,
    param_distributions=param_dist,
    n_iter=50,
    cv=5,
    scoring='accuracy',
    random_state=42,
    n_jobs=-1
)
```

```
random_search.fit(X_train, y_train)
```

Out[7]:



In [8]:
```
print("Best Hyperparameters:", random_search.best_params_)
```

```
Best Hyperparameters: {'colsample_bytree': 0.6467440873590191, 'learning_rate': 0.014223946814525337, 'max_depth': 5, 'n_estimators': 130, 'subsample': 0.855670976374325}
```

In [9]:
```
best_model = random_search.best_estimator_
```

In [10]:
```
y_train_pred = best_model.predict(X_train)
train_accuracy = accuracy_score(y_train, y_train_pred)
print(f"Train Accuracy: {train_accuracy:.3f}")
```

```
Train Accuracy: 0.892
```

In [11]:
```
y_val_pred = best_model.predict(X_val)
accuracy = accuracy_score(y_val, y_val_pred)
print(f"Validation score: {accuracy:.3f}")
```

```
Validation score: 0.827
```

In [12]:
```
test_predictions = best_model.predict(X_test)
```

In [13]:
```
submission = pd.DataFrame({
    "PassengerId": test["PassengerId"],
    "Survived": test_predictions
})

submission.to_csv("submission.csv", index=False)
print("Submission.csv saved.")
```

```
Submission.csv saved.
```