```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
train_df = pd.read_csv(r"C:\Users\Cun\Downloads\titanic1\train.csv")
test_df = pd.read_csv(r"C:\Users\Cun\Downloads\titanic1\test.csv")
```

```python
train_df.columns
```

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

```python
test_df.columns
```

```
Index(['PassengerId', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch',
       'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

```python
train_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

```python
test_df.head()
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

```python
train_df.set_index(train_df.PassengerId, inplace=True)
```

```python
train_df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
PassengerId												
1	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S

```python
train_df.drop('PassengerId', axis =1)
```

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
PassengerId											
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...
887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 11 columns

```
test_df = pd.read_csv(r"C:\Users\Cun\Downloads\titanic1\test.csv", index_col = 'PassengerId')
```

```
test_df.head()
```

	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
PassengerId										
892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

```
#-----------------------------------------------------------------------------------------------------------------------------
```

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 891 entries, 1 to 891
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 90.5+ KB
```

```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 418 entries, 892 to 1309
Data columns (total 10 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Pclass    418 non-null    int64  
 1   Name      418 non-null    object 
 2   Sex       418 non-null    object 
 3   Age       332 non-null    float64
 4   SibSp     418 non-null    int64  
 5   Parch     418 non-null    int64  
 6   Ticket    418 non-null    object 
 7   Fare      417 non-null    float64
 8   Cabin     91 non-null     object 
 9   Embarked  418 non-null    object 
dtypes: float64(2), int64(3), object(5)
```

```
memory usage: 35.9+ KB
```

```
train_df["Survived"] = train_df["Survived"].astype("category")
```

```
train_df["Survived"].dtype
```

```
CategoricalDtype(categories=[0, 1], ordered=False, categories_dtype=int64)
```

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 891 entries, 1 to 891
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   PassengerId  891 non-null    int64   
 1   Survived     891 non-null    category
 2   Pclass       891 non-null    int64   
 3   Name         891 non-null    object  
 4   Sex          891 non-null    object  
 5   Age          714 non-null    float64 
 6   SibSp        891 non-null    int64   
 7   Parch        891 non-null    int64   
 8   Ticket       891 non-null    object  
 9   Fare         891 non-null    float64 
 10  Cabin        204 non-null    object  
 11  Embarked     889 non-null    object  
dtypes: category(1), float64(2), int64(4), object(5)
memory usage: 84.5+ KB
```

```
features = ["Pclass", "Sex", "SibSp", "Parch", "Embarked"]
def convert_cat(df, features):
    for feature in features:
        df[feature] = df[feature].astype("category")
convert_cat(train_df, features)
convert_cat(test_df, features)
```

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 891 entries, 1 to 891
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   PassengerId  891 non-null    int64   
 1   Survived     891 non-null    category
 2   Pclass       891 non-null    category
 3   Name         891 non-null    object  
 4   Sex          891 non-null    category
 5   Age          714 non-null    float64 
 6   SibSp        891 non-null    category
 7   Parch        891 non-null    category
 8   Ticket       891 non-null    object  
 9   Fare         891 non-null    float64 
 10  Cabin        204 non-null    object  
 11  Embarked     889 non-null    category
dtypes: category(6), float64(2), int64(1), object(3)
memory usage: 55.1+ KB
```

```
train_df.describe (include=['category'])
```

	Survived	Pclass	Sex	SibSp	Parch	Embarked
count	891	891	891	891	891	889
unique	2	3	2	7	7	3
top	0	3	male	0	0	S
freq	549	491	577	608	678	644

```
train_df["Survived"].value_counts().to_frame
```

```
<bound method Series.to_frame of Survived
0    549
1    342
Name: count, dtype: int64>
```

```
train_df["Survived"].value_counts(normalize=True).to_frame()
```

	proportion
Survived	
0	0.616162
1	0.383838

```
train_df["Sex"].value_counts().to_frame()
```

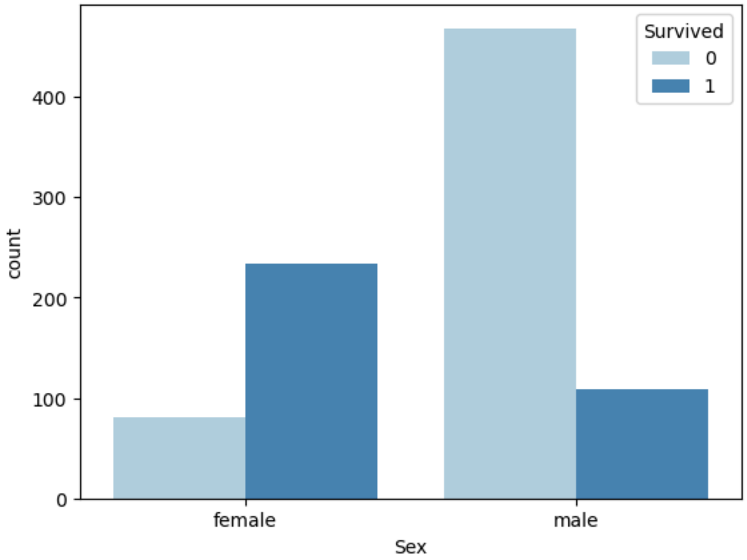
	count
Sex	
male	577
female	314

```
train_df["Sex"].value_counts(normalize=True).to_frame()
```

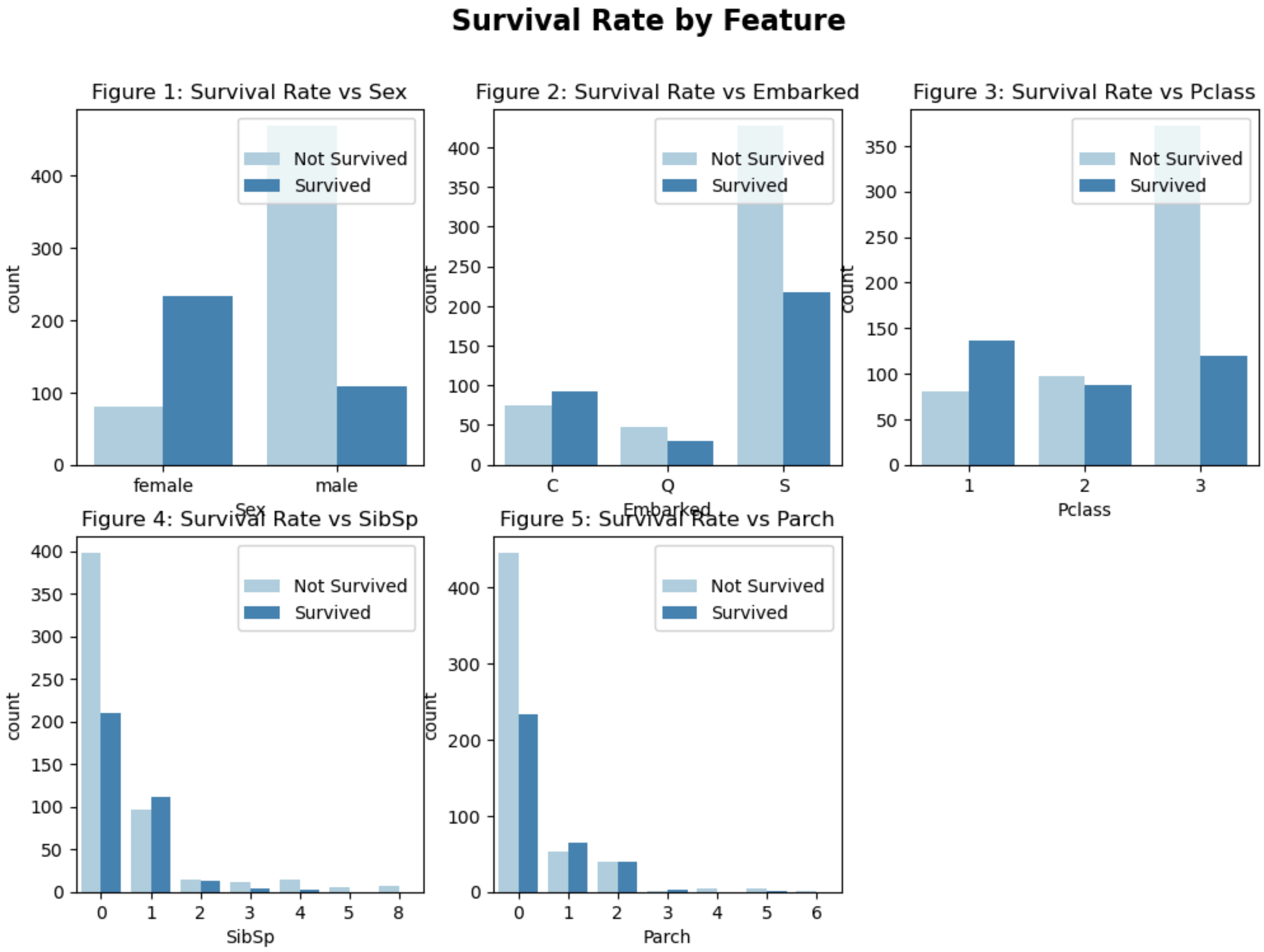
	proportion
Sex	
male	0.647587
female	0.352413

```
sns.countplot(data=train_df, x='Sex', hue='Survived', palette='Blues')
```

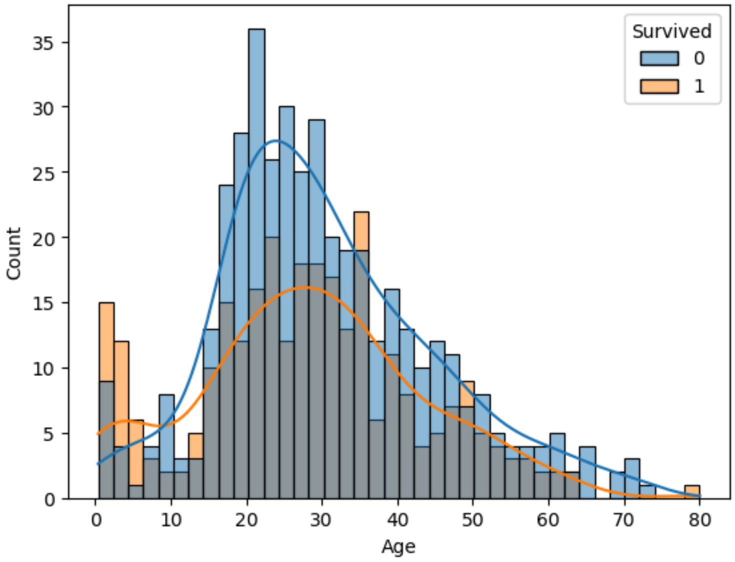
```
<Axes: xlabel='Sex', ylabel='count'>
```



```
cols= ['Sex', 'Embarked', 'Pclass','SibSp', 'Parch']
n_rows = 2
n_cols = 3

fig, ax = plt.subplots(n_rows, n_cols, figsize=(n_cols*4, n_rows*4)) # tăng kích thước hình
fig.suptitle("Survival Rate by Feature", fontsize=16, fontweight='bold')  # tiêu đề chính


for r in range (0, n_rows):
    for c in range (0, n_cols):
        i = r*n_cols + c
        if i<len(cols):
            ax_i = ax[r,c]
            sns.countplot(data= train_df, x=cols[i], hue="Survived", palette="Blues", ax=ax_i)
            ax_i.set_title(f"Figure {i+1}: Survival Rate vs {cols[i]}")
            ax_i.legend(title=' ', loc='upper right', labels=['Not Survived', 'Survived'])
ax.flat[-1].set_visible(False)
plt.tight_layout
plt.show()
```



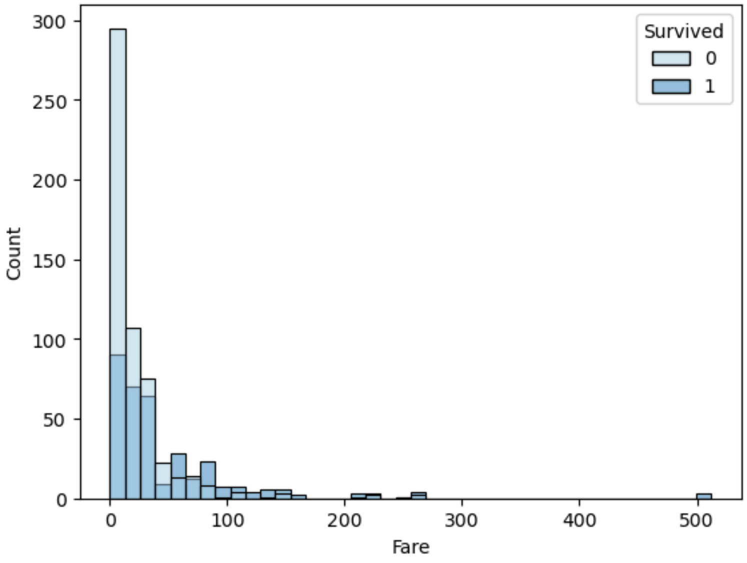
```
sns.histplot(data=train_df, x="Age", hue='Survived', bins = 40, kde=True)
plt.show()
```



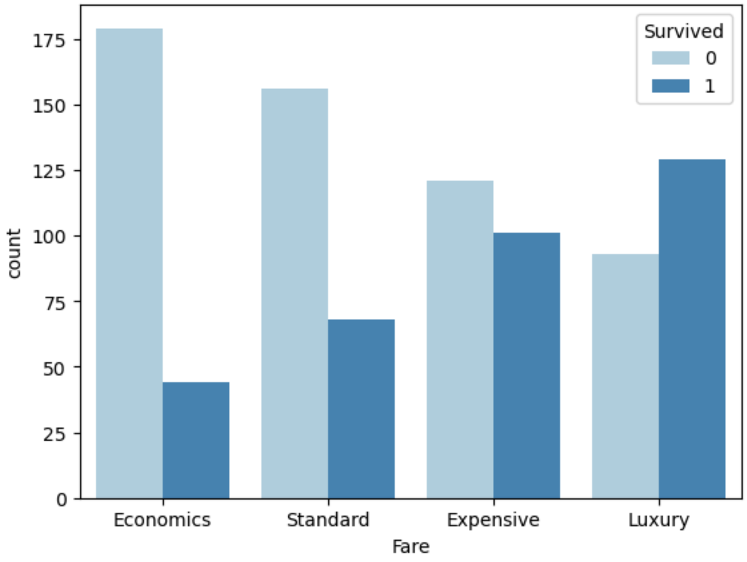
```
train_df["Fare"].describe()
```

```
count    891.000000
mean      32.204208
std       49.693429
min        0.000000
25%        7.910400
50%       14.454200
75%       31.000000
max      512.329200
Name: Fare, dtype: float64
```

```
sns.histplot(data=train_df, x='Fare', hue='Survived', bins=40, palette= 'Blues')
plt.show()
```



```
fare_categories = ['Economics', 'Standard', 'Expensive', 'Luxury']
quartile_data = pd.qcut(train_df['Fare'], 4, labels=fare_categories)

sns.countplot(x=quartile_data, hue=train_df['Survived'], palette="Blues")
plt.show()
```



```
train_df['Name'].head(10)
```

```
PassengerId
1                               Braund, Mr. Owen Harris
2     Cumings, Mrs. John Bradley (Florence Briggs Th...
3                                Heikkinen, Miss. Laina
4          Futrelle, Mrs. Jacques Heath (Lily May Peel)
5                              Allen, Mr. William Henry
6                                      Moran, Mr. James
7                               McCarthy, Mr. Timothy J
8                        Palsson, Master. Gosta Leonard
9     Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
10                  Nasser, Mrs. Nicholas (Adele Achem)
Name: Name, dtype: object
```

```
import re
def extract_title(name):
    p = re.compile(r",([\w\s]+)\.")
    return p.search(name).groups(1)[0].strip()

train_df['Title'] = train_df['Name'].apply(lambda name: extract_title(name))
train_df['Title'].value_counts()
```

```
Title
Mr                517
Miss              182
Mrs               125
Master             40
```

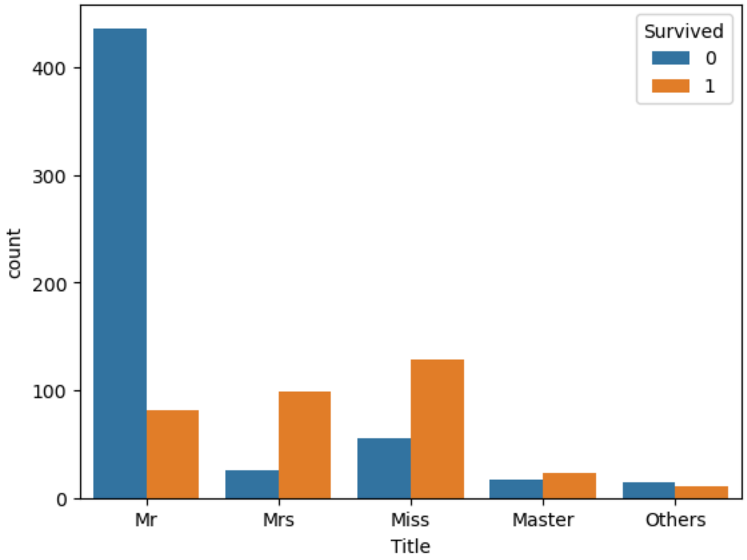
```
Dr              7
Rev             6
Mlle            2
Major           2
Col             2
the Countess    1
Capt            1
Ms              1
Sir             1
Lady            1
Mme             1
Don             1
Jonkheer        1
Name: count, dtype: int64
```

```
test_df['Title'] = test_df['Name'].apply(lambda name: extract_title(name))
test_df['Title'].value_counts()
```

```
Title
Mr        240
Miss       78
Mrs        72
Master     21
Col         2
Rev         2
Ms          1
Dr          1
Dona        1
Name: count, dtype: int64
```

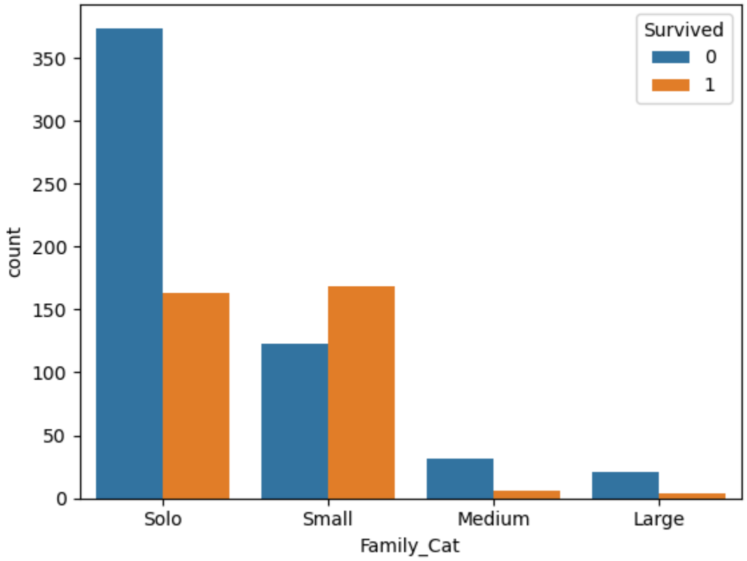
```
def group_title (title):
    if title in ['Mr','Mrs','Miss','Master']:
        return title
    elif title == "Ms":
        return "Miss"
    else:
        return "Others"
train_df['Title'] = train_df['Title'].apply(lambda title: group_title(title))
test_df['Title'] = test_df['Title'].apply(lambda title: group_title(title))
```

```
sns.countplot(data=train_df, x='Title', hue='Survived')
plt.show()
```



```
train_df['Family_Size'] = train_df['SibSp'].astype('int') + train_df['Parch'].astype('int')+1
test_df['Family_Size'] = test_df['SibSp'].astype('int') + test_df['Parch'].astype('int')+1
train_df['Family_Cat'] = pd.cut(train_df['Family_Size'], bins=[0,1,4,6,20], labels = ['Solo', 'Small', 'Medium', 'Large'])
test_df['Family_Cat'] = pd.cut(train_df['Family_Size'], bins=[0,1,4,6,20], labels = ['Solo', 'Small', 'Medium', 'Large'])
```

```
sns.countplot(data=train_df, x='Family_Cat', hue='Survived')
plt.show()
```



```
#         # Data Wrangling
num_features = ['Age', 'Fare']
cat_features = ['Sex', 'Pclass', 'Embarked', 'Title', 'Family_Cat']
feature_cols = num_features + cat_features
print(feature_cols, '\n')
```

```
['Age', 'Fare', 'Sex', 'Pclass', 'Embarked', 'Title', 'Family_Cat']
```

```
def display_missing(df, feature_cols):
    n_rows = df.shape[0]
    for col in feature_cols:
        missing_count = df[col].isnull().sum()
        if missing_count > 0:
            print(f"{col} has {missing_count* 100/n_rows:.2f}% missing values.")

display_missing(train_df, feature_cols)
```

```
Age has 19.87% missing values.
Embarked has 0.22% missing values.
```

```
display_missing(test_df, feature_cols)
```

```
Age has 20.57% missing values.
Fare has 0.24% missing values.
Family_Cat has 100.00% missing values.
```

```
#age_by_sex_pclass = train_df.groupby(['Sex', 'Pclass']).median()['Age']
age_by_sex_pclass = train_df.groupby(['Sex', 'Pclass'])['Age'].median()
```

```
C:\Users\Cun\AppData\Local\Temp\ipykernel_27120\2357480848.py:2: FutureWarning: The default of observed=False is deprecated and will be changed to True
  age_by_sex_pclass = train_df.groupby(['Sex', 'Pclass'])['Age'].median()
```

```
age_by_sex_pclass
```

```
Sex     Pclass
female  1         35.0
        2         28.0
        3         21.5
male    1         40.0
        2         30.0
        3         25.0
Name: Age, dtype: float64
```

```
train_df['Age'] = train_df.groupby(['Sex', 'Pclass'])['Age'].transform(lambda x: x.fillna(x.median()))
```

```
C:\Users\Cun\AppData\Local\Temp\ipykernel_27120\2707403057.py:1: FutureWarning: The default of observed=False is deprecated and will be changed to True
  train_df['Age'] = train_df.groupby(['Sex', 'Pclass'])['Age'].transform(lambda x: x.fillna(x.median()))
```

```
test_df['Age']  = test_df.groupby(['Sex', 'Pclass'])['Age'].transform(lambda x: x.fillna(x.median()))
```

```
C:\Users\Cun\AppData\Local\Temp\ipykernel_27120\1127986851.py:1: FutureWarning: The default of observed=False is deprecated and will be changed to True
  test_df['Age']  = test_df.groupby(['Sex', 'Pclass'])['Age'].transform(lambda x: x.fillna(x.median()))
```

```
display_missing(train_df, feature_cols)
display_missing(test_df, feature_cols)
```

```
Embarked has 0.22% missing values.
Fare has 0.24% missing values.
Family_Cat has 100.00% missing values.
```

```
X = train_df[feature_cols]
y = train_df['Survived']
```

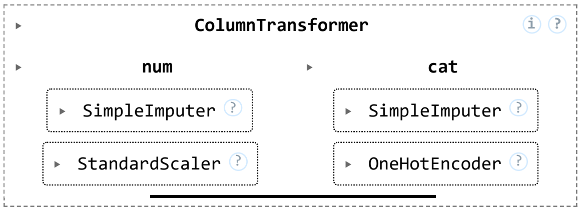
```
X_test = test_df[feature_cols]
```

```
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
```

```
num_transformer = Pipeline(steps =[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

cat_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encode', OneHotEncoder(handle_unknown= 'ignore'))
])
```

```
preprocessor = ColumnTransformer(transformers=
[
    ('num', num_transformer, num_features),
    ('cat', cat_transformer, cat_features)
])
```

```
preprocessor.fit(X)
```



```
X= preprocessor.transform(X)
```

```
X_test = preprocessor.transform(X_test)
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score, classification_report, confusion_matrix
from sklearn.preprocessing import PolynomialFeatures
```

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

```
X_train.shape, X_val.shape
```

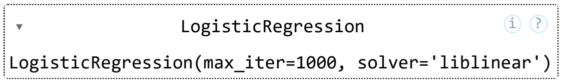
```
((712, 19), (179, 19))
```

```
X_test.shape
```

```
(418, 19)
```

```
from sklearn.linear_model import LogisticRegression
```

```
log_reg = LogisticRegression(solver='liblinear', max_iter=1000)
log_reg.fit(X_train, y_train)
```



```
LogisticRegression(max_iter=1000, solver='liblinear')
```

```
log_reg.score(X_val, y_val)
```

```
0.8044692737430168
```

```
y_pred = log_reg.predict(X_val)
```

```
precision_score(y_val, y_pred), recall_score(y_val, y_pred)
```

```
(0.8208955223880597, 0.7051282051282052)
```

```
print(classification_report(y_val, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.88      0.84       101
           1       0.82      0.71      0.76        78

    accuracy                           0.80       179
   macro avg       0.81      0.79      0.80       179
weighted avg       0.81      0.80      0.80       179
```

```
poly = PolynomialFeatures(degree=5)
poly_features_X_train = poly.fit_transform(X_train)
poly_features_X_val = poly.transform(X_val)
```

```
poly_log_reg = LogisticRegression(solver='liblinear', max_iter=1000)
poly_log_reg.fit(poly_features_X_train, y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000, solver='liblinear')
```

```
poly_log_reg.score(poly_features_X_val, y_val)
```

```
0.7932960893854749
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
decision_tree = DecisionTreeClassifier(criterion = 'entropy', max_depth = 8, random_state=2022)
decision_tree.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=8, random_state=2022)
```

```
decision_tree.score(X_val, y_val)
```

```
0.770949720670391
```

```
#-----------------------------------------------------------------------------------------------------------
```

```
from sklearn.model_selection import cross_val_score
```

```
log_reg_cv = LogisticRegression(solver='liblinear', max_iter = 1000)
dt_cv = DecisionTreeClassifier(criterion = 'entropy', max_depth = 8, random_state=2022)

lr_scores = cross_val_score(log_reg_cv, X, y, scoring='accuracy', cv=5)
```

```
dt_scores = cross_val_score(dt_cv, X, y, scoring='accuracy', cv=5)
```

```
dt_scores.mean(), dt_scores.std()
```

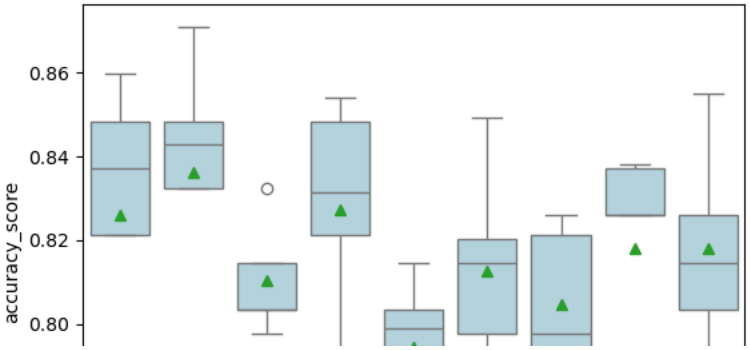
```
(0.8069801016885318, 0.014586754299604428)
```

```
pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\cun\anaconda3\lib\site-packages (3.0.4)
Requirement already satisfied: numpy in c:\users\cun\anaconda3\lib\site-packages (from xgboost) (1.26.4)
Requirement already satisfied: scipy in c:\users\cun\anaconda3\lib\site-packages (from xgboost) (1.13.1)
Note: you may need to restart the kernel to use updated packages.
```

```
from sklearn.svm import LinearSVC, SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, ExtraTreesClassifier, AdaBoostClassifier
from xgboost import XGBClassifier
seed = 2023
models = [
    LinearSVC(max_iter = 12000, random_state=seed),
    SVC (random_state=seed),
    KNeighborsClassifier(metric='minkowski', p=2),
    LogisticRegression(solver='liblinear', max_iter=1000),
    DecisionTreeClassifier(random_state=seed),
    RandomForestClassifier(random_state=seed),
    ExtraTreesClassifier(),
    AdaBoostClassifier(),
    XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=seed)
]


from sklearn.model_selection import StratifiedKFold
def generate_baseline_results(models, X, y, metrics, cv=5, plot_results=False):
# define k-fold:
    kfold = StratifiedKFold(cv, shuffle=True, random_state = seed)
    entries = []
    for model in models:
        model_name = model.__class__.__name__
        #(model_name)
        scores = cross_val_score(model, X,y, scoring=metrics, cv=kfold)
        for fold_idx, score in enumerate(scores):
            entries.append((model_name, fold_idx, score))

    cv_df = pd.DataFrame (entries, columns = ['model_name', 'fold_id', 'accuracy_score'])

    if plot_results:
        sns.boxplot(x='model_name', y='accuracy_score', data= cv_df, color='lightblue', showmeans = True)
        plt.title("Boxplot of Base-Line Model Accuracy using 5-fold cross-validation")
        plt.xticks(rotation = 45)
        plt.show()

    #Summary result:
    mean = cv_df.groupby('model_name')['accuracy_score'].mean()
    std = cv_df.groupby('model_name')['accuracy_score'].std()

    baseline_results = pd.concat([mean, std], axis = 1, ignore_index= True)
    baseline_results.columns = ['Mean', 'Standard Deviation']

    #Sort by accuracy
    baseline_results.sort_values(by=['Mean'], ascending= False, inplace= True)
    return baseline_results
    #return cv_df

generate_baseline_results(models, X, y, metrics = 'accuracy', cv=5, plot_results = True)
```

```
C:\Users\Cun\anaconda3\Lib\site-packages\sklearn\svm\_classes.py:31: FutureWarning: The default value of `dual` will change from `True` to `'auto'` in
  warnings.warn(
C:\Users\Cun\anaconda3\Lib\site-packages\sklearn\svm\_classes.py:31: FutureWarning: The default value of `dual` will change from `True` to `'auto'` in
  warnings.warn(
C:\Users\Cun\anaconda3\Lib\site-packages\sklearn\svm\_classes.py:31: FutureWarning: The default value of `dual` will change from `True` to `'auto'` in
  warnings.warn(
C:\Users\Cun\anaconda3\Lib\site-packages\sklearn\svm\_classes.py:31: FutureWarning: The default value of `dual` will change from `True` to `'auto'` in
  warnings.warn(
C:\Users\Cun\anaconda3\Lib\site-packages\sklearn\svm\_classes.py:31: FutureWarning: The default value of `dual` will change from `True` to `'auto'` in
  warnings.warn(
C:\Users\Cun\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) is deprecated and
  warnings.warn(
C:\Users\Cun\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) is deprecated and
  warnings.warn(
C:\Users\Cun\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) is deprecated and
  warnings.warn(
C:\Users\Cun\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) is deprecated and
  warnings.warn(
C:\Users\Cun\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R algorithm (the default) is deprecated and
  warnings.warn(
C:\Users\Cun\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning: [15:01:03] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
C:\Users\Cun\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning: [15:01:03] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
C:\Users\Cun\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning: [15:01:03] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
C:\Users\Cun\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning: [15:01:03] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
C:\Users\Cun\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning: [15:01:03] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```



Boxplot of Base-Line Model Accuracy using 5-fold cross-validation

```python
from sklearn.svm import SVC
import pandas as pd

# Chọn mô hình tốt nhất (từ bảng bạn đánh giá): SVC
best_model = SVC(kernel="rbf", C=1.0, gamma="scale", random_state=42)

# Train trên toàn bộ train
best_model.fit(X_train, y_train)

# Dự đoán trên test
y_pred = best_model.predict(X_test)

# Xuất submission.csv (418 dòng + header)
submission = pd.DataFrame({
    "PassengerId": test_df.index,   # lấy index thay vì test_df["PassengerId"]
    "Survived": y_pred.astype(int)
})
submission.to_csv("submission.csv", index=False, sep=",")
print("Saved submission.csv")
```

```
Saved submission.csv
```

LogisticRegression	0.827167	0.028974
LinearSVC	0.826044	0.037442