In [5]:
```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\chomo\appdata\local\programs
\python\python312\lib\site-packages (1.5.1)
Requirement already satisfied: numpy>=1.19.5 in c:\users\chomo\appdata\local\program
s\python\python312\lib\site-packages (from scikit-learn) (2.0.0)
Requirement already satisfied: scipy>=1.6.0 in c:\users\chomo\appdata\local\programs
\python\python312\lib\site-packages (from scikit-learn) (1.14.0)
Requirement already satisfied: joblib>=1.2.0 in c:\users\chomo\appdata\local\program
s\python\python312\lib\site-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\chomo\appdata\local
\programs\python\python312\lib\site-packages (from scikit-learn) (3.5.0)
Note: you may need to restart the kernel to use updated packages.
```

```
[notice] A new release of pip is available: 24.0 -> 24.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [10]:
```
import pandas as pd
url = 'https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.cs
titanic_df = pd.read_csv(url)
titanic_df['AgeGroup'] = pd.cut(titanic_df['Age'], bins=[0, 12, 18, 35, 60, 100], l
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
titanic_ml = titanic_df.drop(columns=['Name', 'Ticket', 'Cabin', 'Embarked'])
titanic_ml['Age'] = titanic_ml['Age'].fillna(titanic_ml['Age'].median())
titanic_ml['Fare'] = titanic_ml['Fare'].fillna(titanic_ml['Fare'].median())
label_encoder = LabelEncoder()
titanic_ml['Sex'] = label_encoder.fit_transform(titanic_ml['Sex'])
titanic_ml['AgeGroup'] = label_encoder.fit_transform(titanic_ml['AgeGroup'])
X = titanic_ml.drop(columns=['Survived'])
y = titanic_ml['Survived']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
scaler = StandardScaler()
X_train[['Age', 'Fare']] = scaler.fit_transform(X_train[['Age', 'Fare']])
X_test[['Age', 'Fare']] = scaler.transform(X_test[['Age', 'Fare']])
```

In [11]:
```
print(X_train.head())
print(X_test.head())
print(y_train.head())
print(y_test.head())
```

```
     PassengerId  Pclass  Sex       Age  SibSp  Parch      Fare  AgeGroup
331          332       1    1  1.253641      0      0 -0.078684         3
733          734       2    1 -0.477284      0      0 -0.377145         2
382          383       3    1  0.215086      0      0 -0.474867         2
704          705       3    1 -0.246494      1      0 -0.476230         2
813          814       3    0 -1.785093      4      2 -0.025249         0
     PassengerId  Pclass  Sex       Age  SibSp  Parch      Fare  AgeGroup
709          710       3    1 -0.092634      1      1 -0.333901         5
439          440       2    1  0.138156      0      0 -0.425284         2
840          841       3    1 -0.708074      0      0 -0.474867         2
720          721       2    0 -1.785093      0      1  0.007966         0
39            40       3    0 -1.169653      1      0 -0.411002         1
331    0
733    0
382    0
704    0
813    0
Name: Survived, dtype: int64
709    1
439    0
840    0
720    1
39     1
Name: Survived, dtype: int64
```

In [12]:

```python
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
log_reg = LogisticRegression(solver='lbfgs', max_iter=1000, random_state=42)
decision_tree = DecisionTreeClassifier(random_state=42)
random_forest = RandomForestClassifier(random_state=42)
svc = SVC(random_state=42)
models = {
    'Logistic Regression': log_reg,
    'Decision Tree': decision_tree,
    'Random Forest': random_forest,
    'Support Vector Machine': svc
}

for name, model in models.items():
    # Train the model
    model.fit(X_train, y_train)
    # Make predictions
    y_pred = model.predict(X_test)
    # Evaluate the model
    accuracy = accuracy_score(y_test, y_pred)
    print(f'{name} Accuracy: {accuracy:.4f}')
    print(f'Classification Report for {name}:')
    print(classification_report(y_test, y_pred))
    print('-' * 50)
```

```
Logistic Regression Accuracy: 0.7989
Classification Report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.81      0.87      0.83       105
           1       0.79      0.70      0.74        74

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

--------------------------------------------------
Decision Tree Accuracy: 0.7374
Classification Report for Decision Tree:
              precision    recall  f1-score   support

           0       0.77      0.79      0.78       105
           1       0.69      0.66      0.68        74

    accuracy                           0.74       179
   macro avg       0.73      0.73      0.73       179
weighted avg       0.74      0.74      0.74       179

--------------------------------------------------
Random Forest Accuracy: 0.8101
Classification Report for Random Forest:
              precision    recall  f1-score   support

           0       0.82      0.87      0.84       105
           1       0.79      0.73      0.76        74

    accuracy                           0.81       179
   macro avg       0.81      0.80      0.80       179
weighted avg       0.81      0.81      0.81       179

--------------------------------------------------
Support Vector Machine Accuracy: 0.5866
Classification Report for Support Vector Machine:
              precision    recall  f1-score   support

           0       0.59      1.00      0.74       105
           1       0.00      0.00      0.00        74

    accuracy                           0.59       179
   macro avg       0.29      0.50      0.37       179
weighted avg       0.34      0.59      0.43       179

--------------------------------------------------
```

```
C:\Users\chomo\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\met
rics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and b
eing set to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\chomo\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\met
rics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and b
eing set to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
C:\Users\chomo\AppData\Local\Programs\Python\Python312\Lib\site-packages\sklearn\met
rics\_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined and b
eing set to 0.0 in labels with no predicted samples. Use `zero_division` parameter t
o control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

In [ ]:

```
### Analysis:
#- **Logistic Regression** and **Random Forest** perform relatively well with accur
#and balanced precision and recall scores for both survival classes.
#- **Decision Tree** shows slightly lower accuracy but comparable precision and rec
#- **Support Vector Machine (SVM)** performs poorly in this scenario, showing very
#and precision for predicting survival.

### Recommendations:
#Based on these results, if we prioritize accuracy and balanced performance across
#**Random Forest** appears to be the best-performing model for predicting survival
#It achieves a good balance between precision and recall for both survival classes.
#Further tuning and cross-validation could potentially improve these results,
#especially for models like SVM which performed poorly due to the dataset's charact
```

In [14]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
models = {
    'Logistic Regression': LogisticRegression(solver='lbfgs', max_iter=1000),  # In
    'Decision Tree': DecisionTreeClassifier(),
    'Random Forest': RandomForestClassifier(),
    'SVM': SVC()
}

param_grids = {
    'Logistic Regression': {'C': [0.1, 1]},
    'Decision Tree': {'max_depth': [5, 10], 'min_samples_split': [10]},
    'Random Forest': {'n_estimators': [50], 'max_depth': [5, 10]},
    'SVM': {'C': [0.1, 1], 'kernel': ['linear']}
}

for name, model in models.items():
    grid_search = GridSearchCV(model, param_grids[name], cv=3, scoring='accuracy')
    grid_search.fit(X_train, y_train)
    print(f'Best parameters for {name}: {grid_search.best_params_}')
    print(f'Best cross-validation accuracy for {name}: {grid_search.best_score_:.4f
```

```
Best parameters for Logistic Regression: {'C': 1}
Best cross-validation accuracy for Logistic Regression: 0.7907
Best parameters for Decision Tree: {'max_depth': 5, 'min_samples_split': 10}
Best cross-validation accuracy for Decision Tree: 0.7935
Best parameters for Random Forest: {'max_depth': 5, 'n_estimators': 50}
Best cross-validation accuracy for Random Forest: 0.8244
Best parameters for SVM: {'C': 0.1, 'kernel': 'linear'}
Best cross-validation accuracy for SVM: 0.7907
```

In [ ]:
```
#Analysis:
#Random Forest achieved the highest cross-validation accuracy of approximately 82%,
#indicating it performed slightly better than the other models after tuning.
#Logistic Regression and Decision Tree also performed well, with accuracies around
#SVM with a linear kernel achieved an accuracy of about 79%, showing improvement fr
#Conclusion:
#Based on these results, Random Forest remains the top-performing model after hyper
#It balances accuracy and computational efficiency, making it suitable for predicti
#Further fine-tuning or ensemble methods could potentially enhance its performance.
```