Dataset Preparation

```
import pandas as pd

# Load the dataset
df = pd.read_csv('Titanic.csv')
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
				Futrelle, Mrs. Jacques Heath (Lily May								

Preprocessing

```
# Drop rows with missing values for simplicity
df = df.dropna(subset=['Age', 'Sex', 'Embarked', 'Fare'])

# Encode categorical variables
df['Sex'] = df['Sex'].map({'male': 1, 'female': 0})
df['Embarked'] = df['Embarked'].astype('category').cat.codes

# Select features and target
X = df[['Pclass', 'Sex', 'Age', 'Fare', 'Embarked']]
y = df['Survived']
sensitive_feature = df['Sex']  # Using gender as sensitive attribute
```

```
/tmp/ipython-input-10-1354465156.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cc
  df['Sex'] = df['Sex'].map({'male': 1, 'female': 0})
/tmp/ipython-input-10-1354465156.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-cc
  df['Embarked'] = df['Embarked'].astype('category').cat.codes
```

Model Training and Evaluation

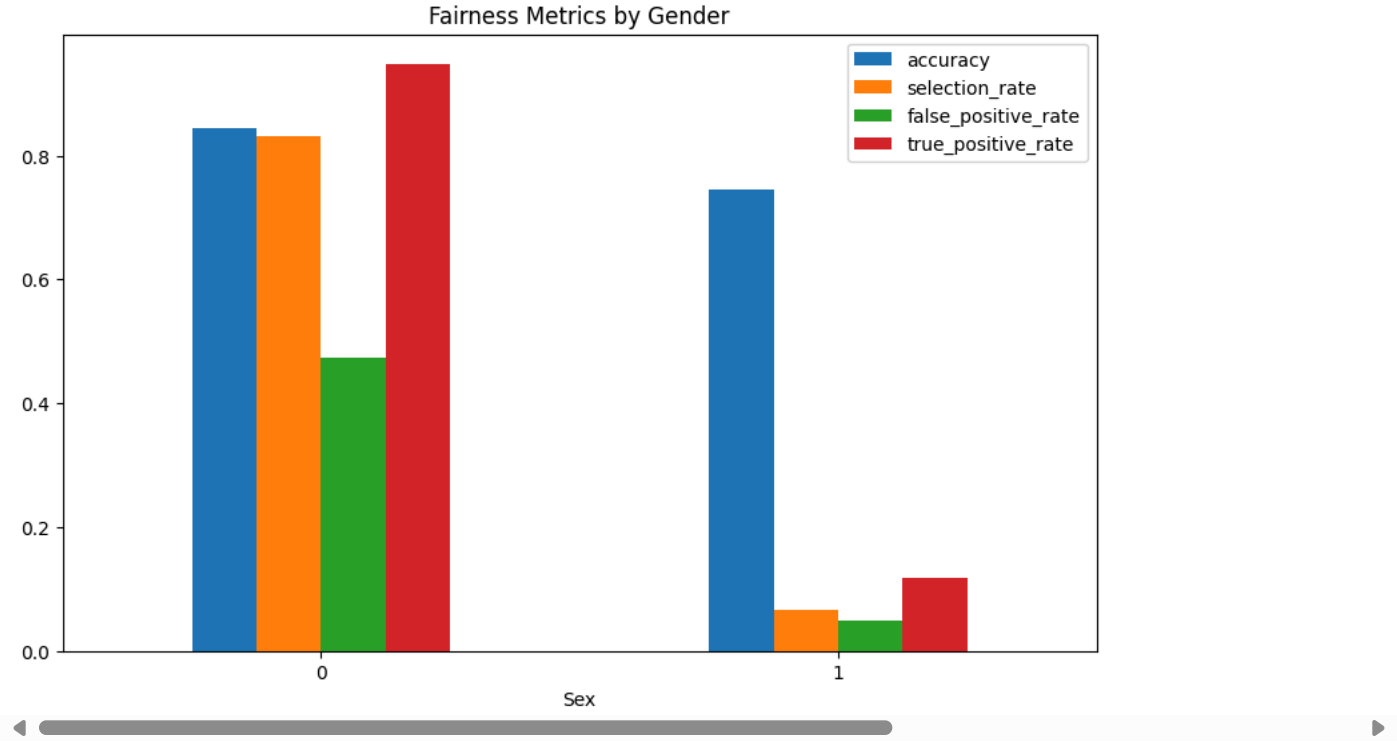
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Split data
X_train, X_test, y_train, y_test, sf_train, sf_test = train_test_split(
    X, y, sensitive_feature, test_size=0.3, random_state=42)

# Train model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.780373831775701
Confusion Matrix:
 [[108  14]
 [ 33  59]]
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.89      0.82       122
           1       0.81      0.64      0.72        92

    accuracy                           0.78       214
   macro avg       0.79      0.76      0.77       214
weighted avg       0.78      0.78      0.78       214
```

Fairness Analysis with Fairlearn

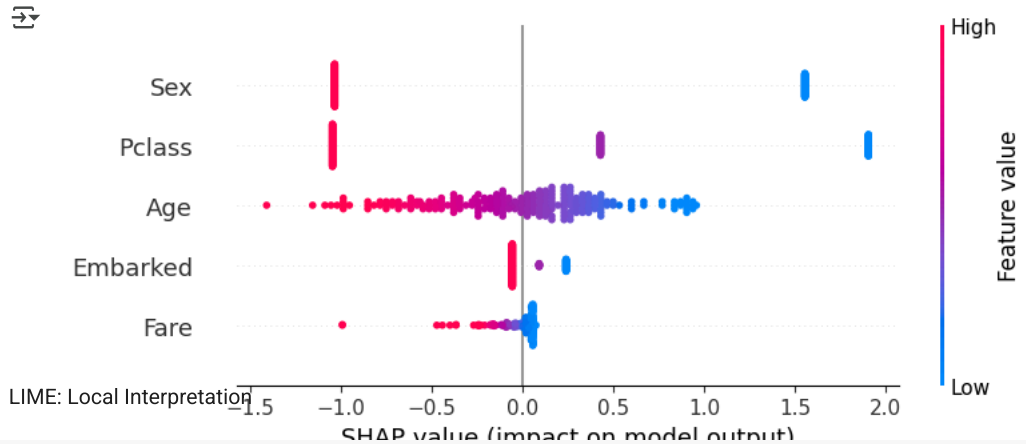
```
!pip install fairlearn --quiet

from fairlearn.metrics import MetricFrame, selection_rate, false_positive_rate, true_positive_rate
import matplotlib.pyplot as plt

# Define metrics
metric_frame = MetricFrame(
    metrics={
        'accuracy': accuracy_score,
        'selection_rate': selection_rate,
        'false_positive_rate': false_positive_rate,
        'true_positive_rate': true_positive_rate
    },
    y_true=y_test,
    y_pred=y_pred,
    sensitive_features=sf_test
)

# Show metrics by group
print(metric_frame.by_group)

# Plot
metric_frame.by_group.plot.bar(figsize=(10, 6), title="Fairness Metrics by Gender")
plt.xticks(rotation=0)
plt.show()
```

```
     accuracy  selection_rate  false_positive_rate  true_positive_rate
Sex
0    0.844156        0.831169             0.473684            0.948276
1    0.744526        0.065693             0.048544            0.117647
```



Explainability Analysis SHAP: Global & Local Explanations

```
!pip install shap --quiet

import shap

explainer = shap.Explainer(model, X_train)
shap_values = explainer(X_test)

# Global summary plot
shap.plots.beeswarm(shap_values)

# Local explanation for an instance
shap.plots.waterfall(shap_values[0])
```



LIME: Local Interpretation

```
!pip install lime --quiet

from lime.lime_tabular import LimeTabularExplainer

lime_explainer = LimeTabularExplainer(
    training_data=X_train.values,
    feature_names=X_train.columns.tolist(),
    class_names=['Not Survived', 'Survived'],
    mode='classification'
)

# Explain a prediction
i = 0  # Change index to explore other predictions
exp = lime_explainer.explain_instance(X_test.iloc[i].values, model.predict_proba)
exp.show_in_notebook()
```

```
69.3 = Fare  −0.07
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 275.7/275.7 kB 5.5 MB/s eta 0:00:00
Preparing metadata (setup.py) ... done
```