titanic_classifications

May 29, 2025

[165]:

```
# Import pandas library
import pandas as pd
# Read csv data file
# Data without feature standardization
df = pd.read_csv('titanic.csv')
```

[166]:

```
# View the number of rows and columns
df.shape
```

[166]:

```
(887, 8)
```

[167]:

```
# View the last 5 rows
df.tail()
```

[167]:

```
     Survived  Pclass                          Name     Sex   Age  \
882         0       2           Rev. Juozas Montvila    male  27.0
883         1       1    Miss. Margaret Edith Graham  female  19.0
884         0       3  Miss. Catherine Helen Johnston  female   7.0
885         1       1           Mr. Karl Howell Behr    male  26.0
886         0       3             Mr. Patrick Dooley    male  32.0

     Siblings/Spouses Aboard  Parents/Children Aboard   Fare
882                        0                        0  13.00
883                        0                        0  30.00
884                        1                        2  23.45
885                        0                        0  30.00
886                        0                        0   7.75
```

[168]:

```
df.dtypes
```

[168]:

```
Survived                     int64
Pclass                       int64
Name                        object
Sex                         object
Age                        float64
Siblings/Spouses Aboard      int64
Parents/Children Aboard      int64
Fare                       float64
```

```
dtype: object
```

[169]:

```
df=df.drop(columns=['Name'])
df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})
df.dtypes
```

[169]:

```
Survived                        int64
Pclass                          int64
Sex                             int64
Age                           float64
Siblings/Spouses Aboard         int64
Parents/Children Aboard         int64
Fare                          float64
dtype: object
```

[170]:

```
df.head()
```

[170]:

```
   Survived  Pclass  Sex   Age  Siblings/Spouses Aboard  \
0         0       3    0  22.0                        1
1         1       1    1  38.0                        1
2         1       3    1  26.0                        0
3         1       1    1  35.0                        1
4         0       3    0  35.0                        0

   Parents/Children Aboard     Fare
0                        0   7.2500
1                        0  71.2833
2                        0   7.9250
3                        0  53.1000
4                        0   8.0500
```

[171]:

```
# Choose features (you can add or remove)
features = ['Pclass', 'Sex', 'Age', 'Siblings/Spouses Aboard', 'Parents/
 ↪Children Aboard', 'Fare']
```

[172]:

```
X = df[features]
y = df['Survived']
```

[173]:

```
from sklearn.model_selection import train_test_split
# Train-test split (80-20)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=666)
```

[174]:

```
from sklearn.tree import DecisionTreeClassifier
# Decision Tree
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
```

```
dt_pred = dt.predict(X_test)
```

[175]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
#Eval of Decision Tree
acc = accuracy_score(y_test, dt_pred)
prec = precision_score(y_test, dt_pred)
rec = recall_score(y_test, dt_pred)
f1 = f1_score(y_test, dt_pred)
cm = confusion_matrix(y_test, dt_pred)

print(f"\n--- Decision Tree ---")
print(f"Accuracy: {acc:.4f}")
print(f"Precision: {prec:.4f}")
print(f"Recall: {rec:.4f}")
print(f"F1 Score: {f1:.4f}")
print("Confusion Matrix:")
print(cm)
```

```

--- Decision Tree ---
Accuracy: 0.7921
Precision: 0.7273
Recall: 0.6452
F1 Score: 0.6838
Confusion Matrix:
[[101  15]
 [ 22  40]]
```

[176]:
```
from sklearn.ensemble import RandomForestClassifier

# Random Forest
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
rf_pred = rf.predict(X_test)
```

[177]:
```
#Eval of RandomForestClassifier
acc = accuracy_score(y_test, rf_pred)
prec = precision_score(y_test, rf_pred)
rec = recall_score(y_test, rf_pred)
f1 = f1_score(y_test, rf_pred)
cm = confusion_matrix(y_test, rf_pred)

print(f"\n--- Random Forest  ---")
print(f"Accuracy: {acc:.4f}")
print(f"Precision: {prec:.4f}")
print(f"Recall: {rec:.4f}")
print(f"F1 Score: {f1:.4f}")
print("Confusion Matrix:")
print(cm)
```

```
--- Random Forest  ---
Accuracy: 0.8202
Precision: 0.7885
Recall: 0.6613
F1 Score: 0.7193
Confusion Matrix:
[[105  11]
 [ 21  41]]
```

[178]:

```
from sklearn.preprocessing import StandardScaler, LabelEncoder
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[179]:

```
from sklearn.svm import SVC
# SVM (scaled data)
svm = SVC(random_state=42)
svm.fit(X_train_scaled, y_train)
svm_pred = svm.predict(X_test_scaled)
```

[180]:

```
#Eval of SVM
acc = accuracy_score(y_test, svm_pred)
prec = precision_score(y_test, svm_pred)
rec = recall_score(y_test, svm_pred)
f1 = f1_score(y_test, svm_pred)
cm = confusion_matrix(y_test, svm_pred)

print(f"\n--- SVM  ---")
print(f"Accuracy: {acc:.4f}")
print(f"Precision: {prec:.4f}")
print(f"Recall: {rec:.4f}")
print(f"F1 Score: {f1:.4f}")
print("Confusion Matrix:")
print(cm)
```

```
--- SVM  ---
Accuracy: 0.8146
Precision: 0.7636
Recall: 0.6774
F1 Score: 0.7179
Confusion Matrix:
[[103  13]
```

```
 [ 20  42]]
```

[181]:

```python
from sklearn.linear_model import LogisticRegression
# Logistic Regression (scaled data)
lr = LogisticRegression(random_state=42, max_iter=1000)
lr.fit(X_train_scaled, y_train)
lr_pred = lr.predict(X_test_scaled)
```

[182]:

```python
#Eval of LR
acc = accuracy_score(y_test, lr_pred)
prec = precision_score(y_test, lr_pred)
rec = recall_score(y_test, lr_pred)
f1 = f1_score(y_test, lr_pred)
cm = confusion_matrix(y_test, lr_pred)

print(f"\n--- LR ---")
print(f"Accuracy: {acc:.4f}")
print(f"Precision: {prec:.4f}")
print(f"Recall: {rec:.4f}")
print(f"F1 Score: {f1:.4f}")
print("Confusion Matrix:")
print(cm)
```

```

--- LR ---
Accuracy: 0.7921
Precision: 0.6984
Recall: 0.7097
F1 Score: 0.7040
Confusion Matrix:
[[97 19]
 [18 44]]
```

[183]:

```python
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
# Linear Discriminant Analysis/Latent Dirichlet Allocation (scaled data)
lda = LinearDiscriminantAnalysis()
lda.fit(X_train_scaled, y_train)
lda_pred = lda.predict(X_test_scaled)
```

[184]:

```python
#Eval of LDA
acc = accuracy_score(y_test, lda_pred)
prec = precision_score(y_test, lda_pred)
rec = recall_score(y_test, lda_pred)
f1 = f1_score(y_test, lda_pred)
cm = confusion_matrix(y_test, lda_pred)

print(f"\n--- LDA  ---")
print(f"Accuracy: {acc:.4f}")
print(f"Precision: {prec:.4f}")
print(f"Recall: {rec:.4f}")
print(f"F1 Score: {f1:.4f}")
print("Confusion Matrix:")
print(cm)
```

```
--- LDA  ---
Accuracy: 0.7753
Precision: 0.6774
Recall: 0.6774
F1 Score: 0.6774
Confusion Matrix:
[[96 20]
 [20 42]]
```

[185]:

```
#WHY SKIPP NAIVE BAYES ?????
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[186]:

```
#Naive Bayes assumes that all features are conditionally independent given the␣
 ↪target.
#Clearly Sex and Survival related, women and child onboard first > Men was left␣
 ↪floating on sea > Jack : "Bitch you aint get no ROSS"
#Wheres Gender Equality when you in dire of surviving amidst ice cold sea @_@?␣
 ↪-JK
#Pclass and Fare related, Rich ASS > Higher Fare (First/Business class >=␣
 ↪Economy)
```

[ ]:

```
```