NATIONAL UNIVERSITY OF MODERN LANGUAGES

ISLAMABAD

Data Mining (LAB)

Lab Quiz: 03

Submitted to
Dr. Moiz-ullah Ghouri

Submitted By
Junaid Asif
(BSAI-144)

Submission Date: December 10, 2024

Train 3 basic models on titanic datasets and compare their confusion metrix, recall, precision, fbeta_score

1. Linear Regression

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import fbeta_score
from sklearn.metrics import classification_report
```

```
# I have 2 files, one is for training dataset and one is for testing dataset
train = pd.read_csv('train_cleaned_data.csv')
test = pd.read_csv('test_cleaned_data.csv')
```

```
features = train.columns.drop('Survived')
X = train[features]
y = train['Survived']
```

```
# Train the model
model = LogisticRegression()
model.fit(X, y)
```

```
C:\Users\junai\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression()
```

```
test['Survived'] = train['Survived']
```

```
test.head()
```

	PassengerId	Pclass	Age	SibSp	Parch	Fare	Gender_male	Embarked_Q	Embarked_S	Survived
0	892	3	2	0	0	0	1	1	0	0
1	893	3	2	1	0	0	0	0	1	1
2	894	2	3	0	0	1	1	1	0	1
3	895	3	2	0	0	1	1	0	1	1
4	896	3	2	1	1	1	0	0	1	0

```
# pridict the model
features = test.columns.drop('Survived')
X_test = test[features]
y_test = test['Survived']
y_pred = model.predict(X_test)
```

```
# confusion matrix
cm = confusion_matrix(y_test, y_pred)
print('Confusion matrix:', cm)
```

```
Confusion matrix: [[152 103]
 [102  61]]
```

```
# Recall
recall = recall_score(y_test, y_pred)
print('Recall:', recall)
```

```
Recall: 0.37423312883435583
```

```
# Precision
precision = precision_score(y_test, y_pred)
print('Precision:', precision)
```

```
Precision: 0.3719512195121951
```

```
# Fbeta_score
fbeta = fbeta_score(y_test, y_pred, beta=0.5)
print('Fbeta_score:', fbeta)
```

```
Fbeta_score: 0.3724053724053724
```

2. KNN

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X, y)
y_pred = model.predict(X_test)
```

Generate + Code + Markdown

```
# confusion matrix
cm = confusion_matrix(y_test, y_pred)
print('Confusion matrix:', cm)
```

```
Confusion matrix: [[255   0]
 [163   0]]
```

```
# Recall
recall = recall_score(y_test, y_pred)
print('Recall:', recall)
```

```
Recall: 0.0
```

```
# Precision
precision = precision_score(y_test, y_pred)
print('Precision:', precision)
```

```
Precision: 0.0
```

```
# Fbeta_score
fbeta = fbeta_score(y_test, y_pred, beta=0.5)
print('Fbeta_score:', fbeta)
```

```
Fbeta_score: 0.0
```

3. SVM

```
from sklearn.svm import SVC
model = SVC()
model.fit(X, y)
y_pred = model.predict(X_test)
```

```
# confusion matrix
cm = confusion_matrix(y_test, y_pred)
print('Confusion matrix:', cm)
```

```
Confusion matrix: [[255   0]
 [163   0]]
```

```
# Recall
recall = recall_score(y_test, y_pred)
print('Recall:', recall)
```

```
Recall: 0.0
```

```
# Precision
precision = precision_score(y_test, y_pred)
print('Precision:', precision)
```

```
Precision: 0.0
```

```
# Fbeta_score
fbeta = fbeta_score(y_test, y_pred, beta=0.5)
print('Fbeta_score:', fbeta)
```

```
Fbeta_score: 0.0
```