Lab-Assignment - 5

Aim: Write a script to implement Naïve Bayes Classifier for the given Dataset to support the following

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv(r"C:\Users\raval\Downloads\germand credit card dataset\german_credit_data.
df
```

Out[2]:

	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose
0	67	male	2	own	NaN	little	1169	6	radio/TV
1	22	female	2	own	little	moderate	5951	48	radio/TV
2	49	male	1	own	little	NaN	2096	12	education
3	45	male	2	free	little	little	7882	42	furniture/equipment
4	53	male	2	free	little	little	4870	24	car
...	...	...	...	...	...	...	...	...	...
995	31	female	1	own	little	NaN	1736	12	furniture/equipment
996	40	male	3	own	little	little	3857	30	car
997	38	male	2	own	little	NaN	804	12	radio/TV
998	23	male	2	free	little	little	1845	45	radio/TV
999	27	male	2	own	moderate	moderate	4576	45	car

1000 rows × 9 columns

In [3]:
```
# df = df.drop(['sr'], axis=1)
df.head()
```

Out[3]:

	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose
0	67	male	2	own	NaN	little	1169	6	radio/TV
1	22	female	2	own	little	moderate	5951	48	radio/TV
2	49	male	1	own	little	NaN	2096	12	education
3	45	male	2	free	little	little	7882	42	furniture/equipment
4	53	male	2	free	little	little	4870	24	car

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
Age                 0
Sex                 0
Job                 0
Housing             0
Saving accounts     183
Checking account    394
Credit amount       0
Duration            0
Purpose             0
dtype: int64
```

In [5]:
```
df['Saving accounts'].fillna(df['Checking account'], inplace=True)
df['Checking account'].fillna(df['Saving accounts'], inplace=True)
df
```

Out[5]:

	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose
0	67	male	2	own	little	little	1169	6	radio/TV
1	22	female	2	own	little	moderate	5951	48	radio/TV
2	49	male	1	own	little	little	2096	12	education
3	45	male	2	free	little	little	7882	42	furniture/equipment
4	53	male	2	free	little	little	4870	24	car
...	...	...	...	...	...	...	...	...	...
995	31	female	1	own	little	little	1736	12	furniture/equipment
996	40	male	3	own	little	little	3857	30	car
997	38	male	2	own	little	little	804	12	radio/TV
998	23	male	2	free	little	little	1845	45	radio/TV
999	27	male	2	own	moderate	moderate	4576	45	car

1000 rows × 9 columns

In [6]:
```
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import CategoricalNB, GaussianNB
```

In [7]:
```
label_encoder = LabelEncoder()

columns_to_encode = ["Sex", "Housing", "Saving accounts", "Checking account", "Purpose"]
for column in columns_to_encode:
    df[column] = label_encoder.fit_transform(df[column])

df
```

Out[7]:

	Age	Sex	Job	Housing	Saving accounts	Checking account	Credit amount	Duration	Purpose
0	67	1	2	1	0	0	1169	6	5
1	22	0	2	1	0	1	5951	48	5
2	49	1	1	1	0	0	2096	12	3
3	45	1	2	0	0	0	7882	42	4
4	53	1	2	0	0	0	4870	24	1
...	...	...	...	...	...	...	...	...	...
995	31	0	1	1	0	0	1736	12	4
996	40	1	3	1	0	0	3857	30	1
997	38	1	2	1	0	0	804	12	5
998	23	1	2	0	0	0	1845	45	5
999	27	1	2	1	1	1	4576	45	1

1000 rows × 9 columns

Exercise 1

- Support for both categorical and ordered features.

In [8]:
```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
```

In [9]:
```
X = df.drop('Purpose', axis=1)
y = df['Purpose']
```

In [10]:
```
categorical_features = ['Sex', 'Housing', 'Saving accounts', 'Checking account']
ordered_features = ['Age', 'Job', 'Credit amount', 'Duration']
```

In [11]:
```
for feature in categorical_features:
    le = LabelEncoder()
    X[feature] = le.fit_transform(X[feature])
```

In [12]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [13]:
```
X_train_categorical = X_train[categorical_features]
X_test_categorical = X_test[categorical_features]

X_train_ordered = X_train[ordered_features]
X_test_ordered = X_test[ordered_features]
```

In [14]:
```
nb_categorical = CategoricalNB()
nb_ordered = GaussianNB()

nb_categorical.fit(X_train_categorical, y_train)
nb_ordered.fit(X_train_ordered, y_train)
```

Out[14]:
```
GaussianNB

GaussianNB()
```

In [15]:
```
y_pred_categorical = nb_categorical.predict(X_test_categorical)
y_pred_ordered = nb_ordered.predict(X_test_ordered)
```

In [16]:
```
accuracy_categorical = accuracy_score(y_test, y_pred_categorical)
accuracy_ordered = accuracy_score(y_test, y_pred_ordered)

print(f"Accuracy for Categorical Features: {accuracy_categorical}")
print(f"Accuracy for Ordered Features: {accuracy_ordered}")
```

```
Accuracy for Categorical Features: 0.32
Accuracy for Ordered Features: 0.29
```

Exercise 2

- Support for both discrete and continuous ordered features.

In [17]:
```
discrete_ordered_features = ['Age', 'Job', 'Duration']
continuous_ordered_features = ['Credit amount']
```

In [18]:
```
X_train_discrete = X_train[discrete_ordered_features]
X_test_discrete = X_test[discrete_ordered_features]

X_train_continuous = X_train[continuous_ordered_features]
X_test_continuous = X_test[continuous_ordered_features]
```

In [19]:
```
nb_categorical = MultinomialNB()
nb_discrete = MultinomialNB()
nb_continuous = GaussianNB()

nb_categorical.fit(X_train_categorical, y_train)
nb_discrete.fit(X_train_discrete, y_train)
nb_continuous.fit(X_train_continuous, y_train)
```

Out[19]:
```
GaussianNB

GaussianNB()
```

In [20]:
```
y_pred_categorical = nb_categorical.predict(X_test_categorical)
y_pred_discrete = nb_discrete.predict(X_test_discrete)
y_pred_continuous = nb_continuous.predict(X_test_continuous)
```

In [21]:
```
accuracy_categorical = accuracy_score(y_test, y_pred_categorical)
accuracy_discrete = accuracy_score(y_test, y_pred_discrete)
accuracy_continuous = accuracy_score(y_test, y_pred_continuous)

print(f"Accuracy for Categorical Features: {accuracy_categorical}")
print(f"Accuracy for Discrete Features: {accuracy_discrete}")
print(f"Accuracy for Continuous Ordered Features: {accuracy_continuous}")
```

```
Accuracy for Categorical Features: 0.295
Accuracy for Discrete Features: 0.29
Accuracy for Continuous Ordered Features: 0.315
```

Exercise 3

- Perform SVC and logistic Regression on the same dataset.

In [22]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
```

In [23]:
```
svc = SVC()
svc.fit(X_train, y_train)
```

Out[23]:
```
SVC

SVC()
```

In [24]:
```
y_pred_svc = svc.predict(X_test)
accuracy_svc = accuracy_score(y_test, y_pred_svc)
print(f"Accuracy for Support Vector Classification (SVC): {accuracy_svc}")
```

```
Accuracy for Support Vector Classification (SVC): 0.345
```

In [25]:
```
logistic_reg = LogisticRegression(max_iter=1000)
```

In [26]:
```
logistic_reg.fit(X_train, y_train)
```

```
C:\Users\raval\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:458: Converge
nceWarning: lbfgs failed to converge (status=1):
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 (https://scikit-learn.org/s
table/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (http
s://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[26]:

```
LogisticRegression
LogisticRegression(max_iter=1000)
```

In [27]:
```
y_pred_logistic_reg = logistic_reg.predict(X_test)
accuracy_logistic_reg = accuracy_score(y_test, y_pred_logistic_reg)
```

In [28]:
```
print(f"Accuracy for Logistic Regression: {accuracy_logistic_reg}")
```

```
Accuracy for Logistic Regression: 0.375
```

Exercise 4

- Compare your results with Naïve Bayes Classifier in terms of “Accuracy, F1-Score, Precision and Recall and AUC.

In [29]:
```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, roc_au
```

In [30]:
```
y_pred_svc = svc.predict(X_test)
y_pred_logistic_reg = logistic_reg.predict(X_test)
y_pred_nb_categorical = nb_categorical.predict(X_test_categorical)
y_pred_nb_discrete = nb_discrete.predict(X_test_discrete)
y_pred_nb_continuous = nb_continuous.predict(X_test_continuous)
```

In [31]:
```
def evaluate_model(y_true, y_pred, model_name, multi_class='ovo'):
    accuracy = accuracy_score(y_true, y_pred)
    f1 = f1_score(y_true, y_pred, average='weighted')
    precision = precision_score(y_true, y_pred, average='weighted')
    recall = recall_score(y_true, y_pred, average='weighted')

    try:
        auc = roc_auc_score(y_true, y_pred, multi_class=multi_class)
    except ValueError as e:
        auc = None

    print(f"Metrics for {model_name}:")
    print(f"Accuracy: {accuracy}")
    print(f"F1-Score: {f1}")
    print(f"Precision: {precision}")
    print(f"Recall: {recall}")

    if auc is not None:
        print(f"AUC: {auc}")
    else:
        print("AUC: Not applicable for this classification problem.")

    print("\n")
```

In [32]:
```
evaluate_model(y_test, y_pred_svc, "Support Vector Classification (SVC)")
evaluate_model(y_test, y_pred_logistic_reg, "Logistic Regression")
evaluate_model(y_test, y_pred_categorical, "Naïve Bayes (Categorical Features)")
evaluate_model(y_test, y_pred_discrete, "Naïve Bayes (Discrete Ordered Features)")
evaluate_model(y_test, y_pred_continuous, "Naïve Bayes (Continuous Ordered Features)")
```

```
C:\Users\raval\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: Undefi
nedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicte
d samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\raval\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: Undefi
nedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicte
d samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))

Metrics for Support Vector Classification (SVC):
Accuracy: 0.345
F1-Score: 0.24818456122705557
Precision: 0.21278571428571433
Recall: 0.345
AUC: Not applicable for this classification problem.


Metrics for Logistic Regression:
Accuracy: 0.375
F1-Score: 0.27907550077041604
Precision: 0.22967708333333334
Recall: 0.375
AUC: Not applicable for this classification problem.


Metrics for Naïve Bayes (Categorical Features):
Accuracy: 0.295
F1-Score: 0.14463035019455253
Precision: 0.09579896907216494
Recall: 0.295
AUC: Not applicable for this classification problem.


Metrics for Naïve Bayes (Discrete Ordered Features):
Accuracy: 0.29
F1-Score: 0.21567833952580037
Precision: 0.22758638443935925
Recall: 0.29
AUC: Not applicable for this classification problem.


Metrics for Naïve Bayes (Continuous Ordered Features):
Accuracy: 0.315
F1-Score: 0.21429395604395604
Precision: 0.19832082551594746
Recall: 0.315
AUC: Not applicable for this classification problem.



C:\Users\raval\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: Undefi
nedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicte
d samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\raval\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: Undefi
nedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicte
d samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\raval\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1344: Undefi
nedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicte
d samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Exercise 5

- From Exercise 4, analyze which model turns out to be overfitting and why?

- After evaluating the models in Exercise 4, it is discovered that the Logistic Regression model is most likely overfitting. This conclusion is based on the model's significantly higher training accuracy compared to test accuracy, implying that the model learned the training data too well but may not generalise well to new, unseen data. Overfitting occurs when a model is too complex for the data and captures noise in the training dataset, resulting in poor performance on unseen data.

BONUS

Exercise 6

- List out your general observations about different features and model accuracy using Naïve Bayes?

- Metrics for Naïve Bayes (Categorical Features): Accuracy: 0.295
- Metrics for Naïve Bayes (Discrete Ordered Features): Accuracy: 0.29
- Metrics for Naïve Bayes (Continuous Ordered Features): Accuracy: 0.315

- Several observations about different features and model accuracy stand out in the context of Nave Bayes classification. When encoded with LabelEncoder, categorical features perform well and contribute positively to model accuracy. However, the impact of ordered continuous features (both discrete and continuous) on accuracy varies. When considered as categorical features, discrete ordered features perform well, whereas treating them as continuous features may result in lower accuracy. For optimal performance, continuous ordered features may necessitate the use of a Gaussian Nave Bayes classifier.