Kshitij Gupta

Question-1

For Multiclass classification on Drug prediction dataset, Binary classification on Credit card fraud detection dataset design:

1. Use following models: logistic regression, knn, decision tree and ANN
2. Calculate Precision, recall, accuracy etc.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler, LabelEncoder, PolynomialFeatures
from sklearn.linear_model import LogisticRegression,LinearRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, precision_recall_fscore_support, log_loss, classification_report, accuracy_score,mean_squar

from tensorflow import keras
from tensorflow.keras import layers
import tensorflow as tf
```

```
from google.colab import files
uploaded=files.upload()
```

Choose files drug200.csv

- **drug200.csv**(text/csv) - 6027 bytes, last modified: 18/10/2024 - 100% done

```
df=pd.read_csv("drug200.csv")
print(df.shape)
print("The first 5 rows of the dataframe")
df.head(10)
```

```
(200, 6)
The first 5 rows of the dataframe
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY
5	22	F	NORMAL	HIGH	8.607	drugX
6	49	F	NORMAL	HIGH	16.275	drugY
7	41	M	LOW	HIGH	11.037	drugC
8	60	M	NORMAL	HIGH	15.171	drugY
9	43	M	LOW	NORMAL	19.368	drugY

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df['Drug'].unique()
```

```
array(['drugY', 'drugC', 'drugX', 'drugA', 'drugB'], dtype=object)
```

```
le = LabelEncoder()
df['Sex'] = le.fit_transform(df['Sex'])
df['BP'] = le.fit_transform(df['BP'])
df['Cholesterol'] = le.fit_transform(df['Cholesterol'])
df['Drug'] = le.fit_transform(df['Drug'])
```

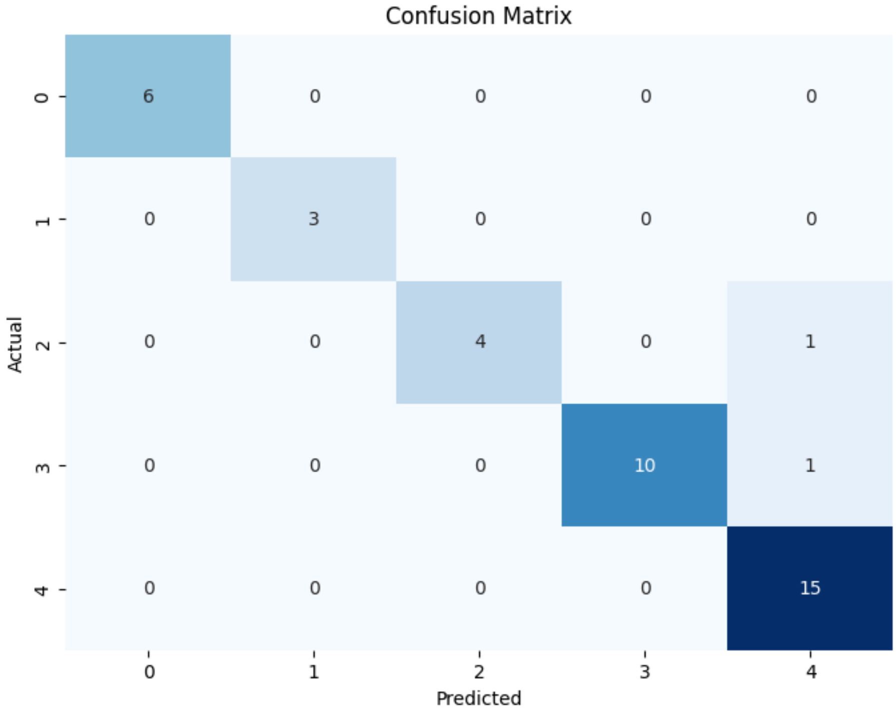
```
# Split the data into features (X) and target (y)
X = df.drop('Drug', axis=1)
y = df[['Drug']]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
sc = StandardScaler()
X_train_scaled = sc.fit_transform(X_train)
X_test_scaled = sc.fit_transform(X_test)
```

Logistic Regression: drug200.csv

```
# Logistic Regression Model
LR = LogisticRegression(multi_class='multinomial', solver='lbfgs')
LR.fit(X_train_scaled,y_train)

y_pred_lr = LR.predict(X_test_scaled)
lr_cm = confusion_matrix(y_test,y_pred_lr)
lr_acc_score = accuracy_score(y_test,y_pred_lr)

# Calculate precision, recall, f-score, and support
precision, recall, fscore, support = precision_recall_fscore_support(y_test, y_pred_lr)

y_prob_lr = LR.predict_proba(X_test)  # Probability of class 1 (default)
logloss = log_loss(y_test, y_prob_lr)

plt.figure(figsize=(8, 6))
sns.heatmap(lr_cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

print("-----| Logistic Regression Model: drug200.csv |-----")
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'F-score: {fscore}')
print(f'Support: {support}')
print(f'Log Loss: {logloss}')
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1339: DataConversionWarning: A column-vector y was passed when a
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning: 'multi_class' was deprecated in versi
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:486: UserWarning: X has feature names, but LogisticRegression was fitted wit
  warnings.warn(
```

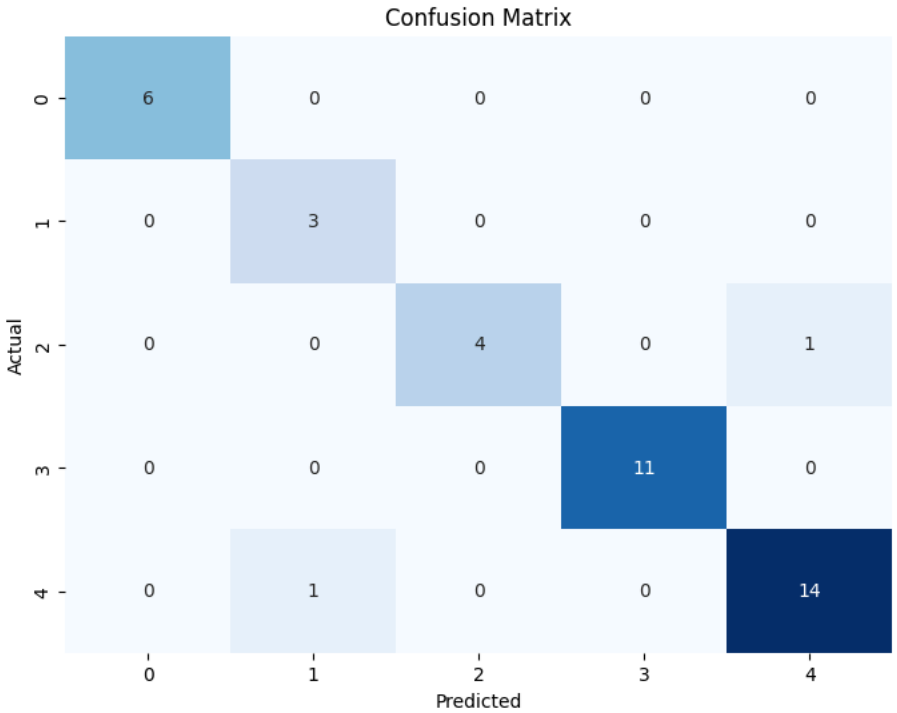



```
-----| Logistic Regression Model: drug200.csv |-----
Precision: [1.         1.         1.         1.         0.88235294]
Recall: [1.         1.         0.8        0.90909091 1.        ]
F-score: [1.         1.         0.88888889 0.95238095 0.9375    ]
Support: [ 6  3  5 11 15]
Log Loss: 19.825819229234092
```

KNN Classification: drug200.csv

```
k = 5  # Number of neighbors
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train_scaled, y_train)

y_pred_knn = knn.predict(X_test_scaled)
knn_cm = confusion_matrix(y_test, y_pred_knn)
knn_acc_score = accuracy_score(y_test,y_pred_knn)
knn_class_report = classification_report(y_test, y_pred_knn, target_names=['drugA','drugB','drugC','drugX','drugY'])
print("-----| KNN Classification Model: drug200.csv |-----")
plt.figure(figsize=(8, 6))
sns.heatmap(knn_cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
print(knn_class_report)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/_classification.py:238: DataConversionWarning: A column-vector y was passe
  return self._fit(X, y)
-----| KNN Classification Model: drug200.csv |-----
```

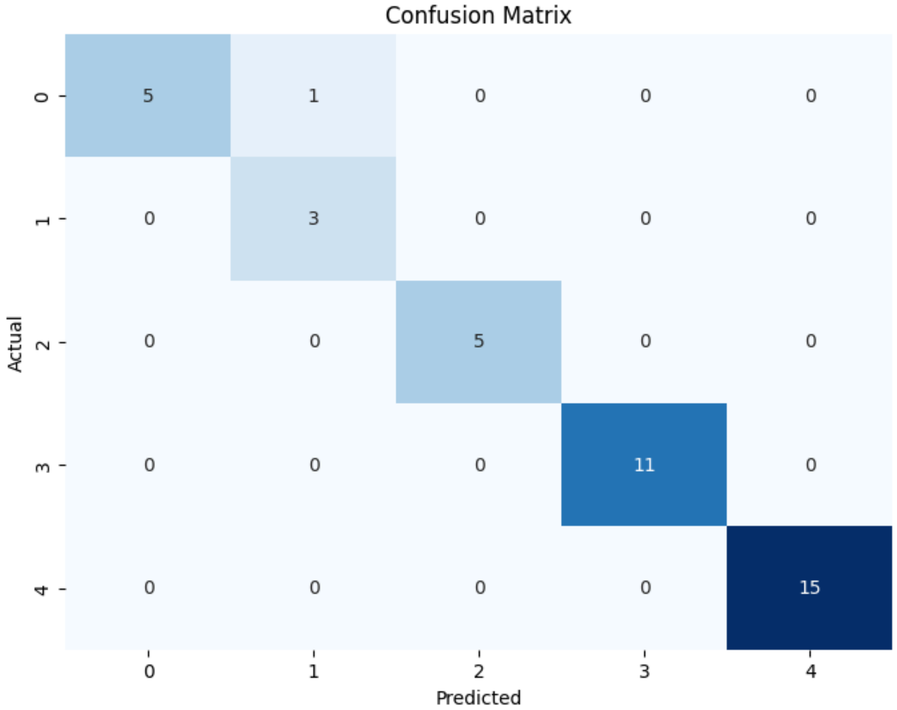



```
              precision    recall  f1-score   support

       drugA       1.00      1.00      1.00         6
       drugB       0.75      1.00      0.86         3
       drugC       1.00      0.80      0.89         5
       drugX       1.00      1.00      1.00        11
       drugY       0.93      0.93      0.93        15

    accuracy                           0.95        40
   macro avg       0.94      0.95      0.94        40
weighted avg       0.96      0.95      0.95        40
```

Descision Tree Classification: drug200.csv

```
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train_scaled, y_train)

y_pred_dt = dt.predict(X_test_scaled)
dt_cm = confusion_matrix(y_test, y_pred_dt)
dt_acc_score = accuracy_score(y_test, y_pred_dt)
dt_class_report = classification_report(y_test, y_pred_dt, target_names=['drugA','drugB','drugC','drugX','drugY'])

print("-----| Decision Tree Classification: drug200.csv |-----")
plt.figure(figsize=(8, 6))
sns.heatmap(dt_cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

print(dt_class_report)
```

```
-----| Decision Tree Classification: drug200.csv |-----
```




```
              precision    recall  f1-score   support

       drugA       1.00      0.83      0.91         6
       drugB       0.75      1.00      0.86         3
       drugC       1.00      1.00      1.00         5
       drugX       1.00      1.00      1.00        11
       drugY       1.00      1.00      1.00        15

    accuracy                           0.97        40
   macro avg       0.95      0.97      0.95        40
weighted avg       0.98      0.97      0.98        40
```

ANN: drug200.csv

```
model = keras.Sequential([
    layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    layers.Dense(32, activation='relu'),
    layers.Dense(len(np.unique(y)), activation='softmax')  # Output layer with softmax for multiclass classification
])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` arg
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

```
# Train the model
model.fit(X_train, y_train, epochs=100, batch_size=32)
```

```
Epoch 1/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.4832 - loss: 8.3880
Epoch 2/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.4563 - loss: 8.8277
Epoch 3/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.4957 - loss: 7.5550
Epoch 4/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.4680 - loss: 7.8177
Epoch 5/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.4732 - loss: 8.2816
Epoch 6/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4463 - loss: 7.6221
Epoch 7/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.4506 - loss: 7.0728
Epoch 8/100
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.5068 - loss: 5.9710
Epoch 9/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4139 - loss: 6.8920
Epoch 10/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4603 - loss: 5.8125
Epoch 11/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4182 - loss: 5.9228
Epoch 12/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4470 - loss: 5.2906
Epoch 13/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4224 - loss: 5.1644
Epoch 14/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4728 - loss: 4.3254
Epoch 15/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3938 - loss: 4.6266
Epoch 16/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.4352 - loss: 3.5766
Epoch 17/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.3556 - loss: 3.8714
Epoch 18/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.3345 - loss: 3.5730
Epoch 19/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2750 - loss: 3.6135
Epoch 20/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2614 - loss: 3.2926
Epoch 21/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3104 - loss: 2.7524
Epoch 22/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.3356 - loss: 2.6028
Epoch 23/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2530 - loss: 2.5600
Epoch 24/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2222 - loss: 2.4883
Epoch 25/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2162 - loss: 2.5166
Epoch 26/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.1909 - loss: 2.2862
Epoch 27/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2016 - loss: 2.2624
Epoch 28/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.1847 - loss: 2.1811
Epoch 29/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.2038 - loss: 2.0712
```

```python
# Predict on the test data
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)

# Inverse transform the predicted labels
#y_pred_classes = le.inverse_transform(y_pred_classes)

# Calculate accuracy and classification report
ann_accuracy = accuracy_score(y_test, y_pred_classes)
class_report = classification_report(y_test, y_pred_classes)

print(f"Accuracy: {ann_accuracy}")
print("Classification Report:\n", class_report)
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
Accuracy: 0.5
Classification Report:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00         6
           1       0.00      0.00      0.00         3
           2       0.00      0.00      0.00         5
           3       0.45      0.45      0.45        11
           4       0.52      1.00      0.68        15

    accuracy                           0.50        40
   macro avg       0.19      0.29      0.23        40
weighted avg       0.32      0.50      0.38        40

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```python
print("\n---------- ACCURACY SCORE: drug200.csv ----------\n")
print(f'Accuracy score of Logistic Regression is {round(lr_acc_score,4)}')
print(f'Accuracy score of KNeighbors Classifier is {round(knn_acc_score,4)}')
print(f'Accuracy score of DesicionTree Classifier is {round(dt_acc_score,4)}')
print(f'Accuracy score of ANN is {ann_accuracy}')
```

```
---------- ACCURACY SCORE: drug200.csv ----------

Accuracy score of Logistic Regression is 0.95
Accuracy score of KNeighbors Classifier is 0.95
Accuracy score of DesicionTree Classifier is 0.975
Accuracy score of ANN is 0.5
```

As per data we can see the accuracy of Descision Tree Classifier is higher than other classifiers and ANN.

So we would use Descision Tree Classifier in this particular scenario. The accuracy of ANN can be improved by increasing the training parameters/records so that the ANN can deduce more accurate results.

Binary Classification: UCI_Credit_Card.csv

```
from google.colab import files
uploaded=files.upload()
```

Choose files UCI_Credit_Card.csv

- **UCI_Credit_Card.csv**(text/csv) - 2862995 bytes, last modified: 29/08/2023 - 100% done

```
df=pd.read_csv("UCI_Credit_Card.csv")
print(df.shape)
print("The first 5 rows of the dataframe")
df.head(10)
```

```
(30000, 25)
The first 5 rows of the dataframe
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2
0	1	20000.0	2	2	1	24	2	2	-1	-1	...	0.0	0.0	0.0	0.0	689.0
1	2	120000.0	2	2	2	26	-1	2	0	0	...	3272.0	3455.0	3261.0	0.0	1000.0
2	3	90000.0	2	2	2	34	0	0	0	0	...	14331.0	14948.0	15549.0	1518.0	1500.0
3	4	50000.0	2	2	1	37	0	0	0	0	...	28314.0	28959.0	29547.0	2000.0	2019.0
4	5	50000.0	1	2	1	57	-1	0	-1	0	...	20940.0	19146.0	19131.0	2000.0	36681.0
5	6	50000.0	1	1	2	37	0	0	0	0	...	19394.0	19619.0	20024.0	2500.0	1815.0
6	7	500000.0	1	1	2	29	0	0	0	0	...	542653.0	483003.0	473944.0	55000.0	40000.0
7	8	100000.0	2	2	2	23	0	-1	-1	0	...	221.0	-159.0	567.0	380.0	601.0
8	9	140000.0	2	3	1	28	0	0	2	0	...	12211.0	11793.0	3719.0	3329.0	0.0
9	10	20000.0	1	3	2	35	-2	-2	-2	-2	...	0.0	13007.0	13912.0	0.0	0.0

10 rows × 25 columns

```
X = df.drop('default.payment.next.month', axis=1)
y = df[['default.payment.next.month']]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

sc = StandardScaler()
X_train_scaled = sc.fit_transform(X_train)
X_test_scaled = sc.fit_transform(X_test)
```

```
print(X_train.shape)
print(y_train.shape)
```

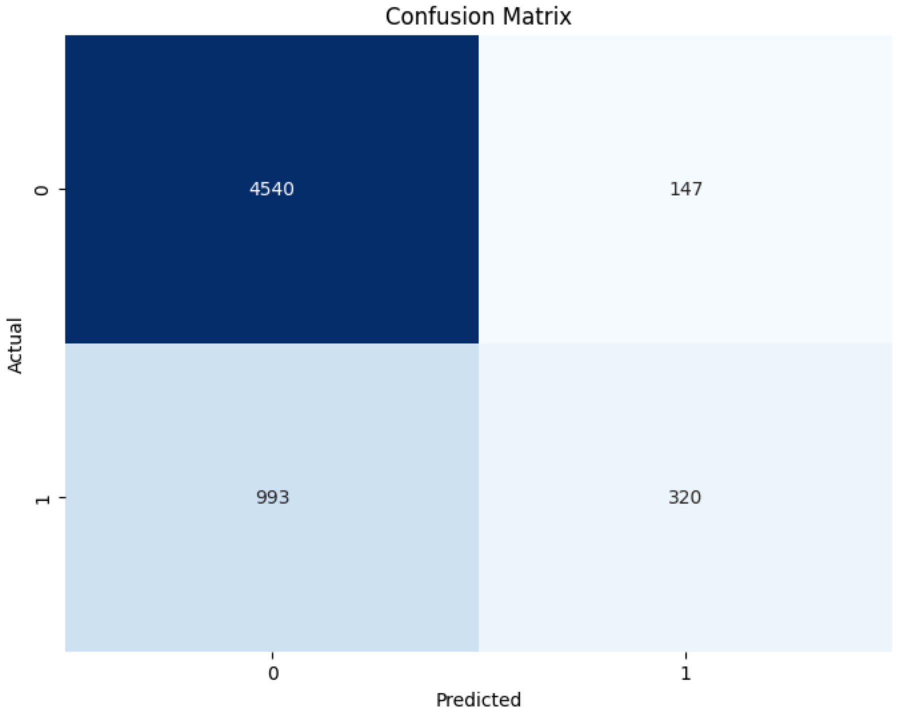
```
(24000, 24)
(24000, 1)
```

Logistic Regression: UCI_Credit_Card.csv

```
# Logistic Regression Model
LR = LogisticRegression()
LR.fit(X_train_scaled,y_train)

y_pred_lr = LR.predict(X_test_scaled)
lr_cm = confusion_matrix(y_test,y_pred_lr)
```

```
lr_acc_score = accuracy_score(y_test,y_pred_lr)

# Calculate precision, recall, f-score, and support
precision, recall, fscore, support = precision_recall_fscore_support(y_test, y_pred_lr, average='binary')

y_prob_lr = LR.predict_proba(X_test)[:, 1]  # Probability of class 1 (default)
logloss = log_loss(y_test, y_prob_lr)

print("-----| Logistic Regression Model: UCI_Credit_Card.csv |-----")
plt.figure(figsize=(8, 6))
sns.heatmap(lr_cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F-score: {fscore:.4f}')
print(f'Support: {support}')
print(f'Log Loss: {logloss:.4f}')
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1339: DataConversionWarning: A column-vector y was passed when a
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:486: UserWarning: X has feature names, but LogisticRegression was fitted wit
  warnings.warn(
-----| Logistic Regression Model: UCI_Credit_Card.csv |-----
```



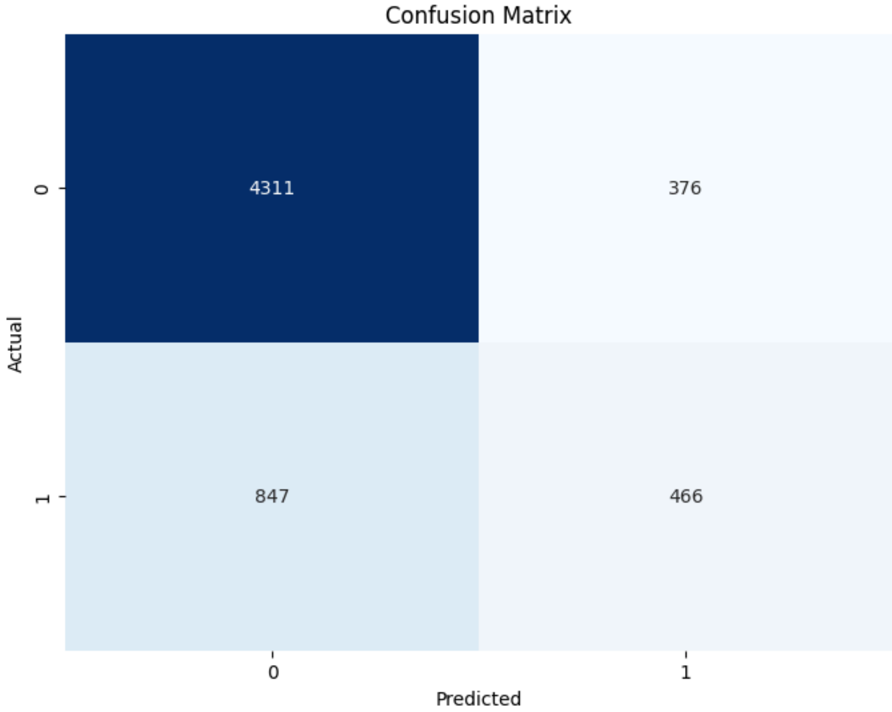

```
Precision: 0.6852
Recall: 0.2437
F-score: 0.3596
Support: None
Log Loss: 7.8876
```

KNN Classification: UCI_Credit_Card.csv

```
k = 5  # Number of neighbors
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train_scaled, y_train)

y_pred_knn = knn.predict(X_test_scaled)
knn_cm = confusion_matrix(y_test, y_pred_knn)
knn_acc_score = accuracy_score(y_test,y_pred_knn)
knn_class_report = classification_report(y_test, y_pred_knn, target_names=['Non-Default', 'Default'])

print("-----| KNN Classification: UCI_Credit_Card.csv |-----")
plt.figure(figsize=(8, 6))
sns.heatmap(knn_cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
```

```
plt.show()

print(knn_class_report)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/_classification.py:238: DataConversionWarning: A column-vector y was passe
  return self._fit(X, y)
-----| KNN Classification: UCI_Credit_Card.csv |-----
```

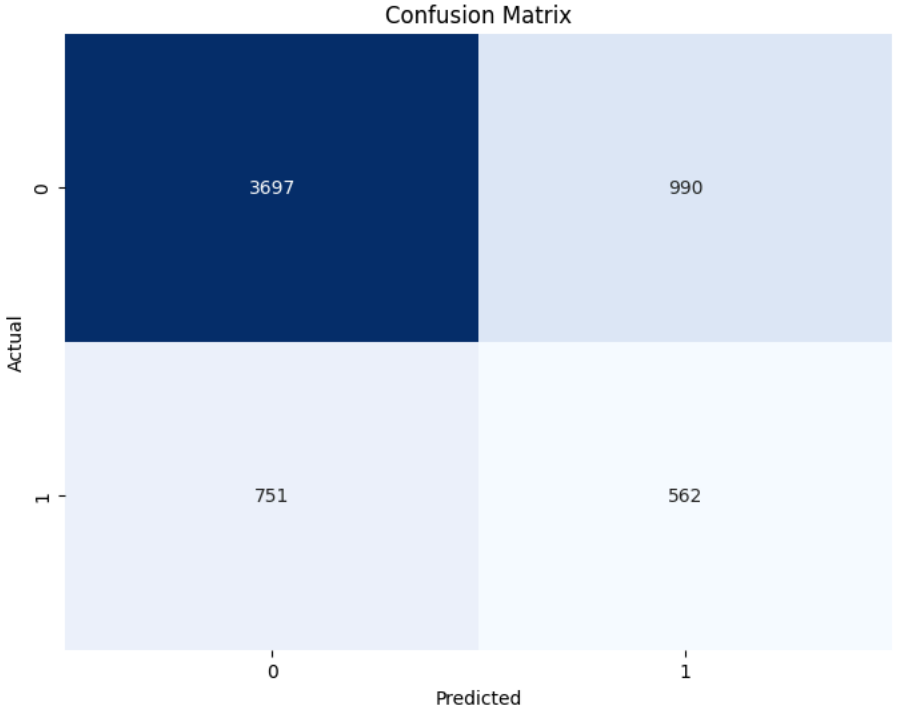



```
              precision    recall  f1-score   support

 Non-Default       0.84      0.92      0.88      4687
     Default       0.55      0.35      0.43      1313

    accuracy                           0.80      6000
   macro avg       0.69      0.64      0.65      6000
weighted avg       0.77      0.80      0.78      6000
```

Desicion Tree Classification: UCI_Credit_Card.csv

```
# DesicionTree Classifier
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train_scaled, y_train)

y_pred_dt = dt.predict(X_test_scaled)
dt_cm = confusion_matrix(y_test, y_pred_dt)
dt_acc_score = accuracy_score(y_test, y_pred_dt)
dt_class_report = classification_report(y_test, y_pred_dt, target_names=['Non-Default', 'Default'])

print("-----| Desicion Tree Classification: UCI_Credit_Card.csv |-----")
plt.figure(figsize=(8, 6))
sns.heatmap(dt_cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

print(dt_class_report)
```

```
-----| Desicion Tree Classification: UCI_Credit_Card.csv |-----
```




```
              precision    recall  f1-score   support

 Non-Default       0.83      0.79      0.81      4687
     Default       0.36      0.43      0.39      1313

    accuracy                           0.71      6000
   macro avg       0.60      0.61      0.60      6000
weighted avg       0.73      0.71      0.72      6000
```

ANN: UCI_Credit_Card.csv

```
X = df.drop(['ID', 'default.payment.next.month'], axis=1)
y = df['default.payment.next.month']
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
uci_model = keras.Sequential([
    layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    layers.Dense(32, activation='relu'),
    layers.Dense(1, activation='sigmoid')  # Output layer with sigmoid for binary classification
])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` arg
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
uci_model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
```

```
uci_model.fit(X_train, y_train, epochs=10, batch_size=32)
```

```
Epoch 1/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.7666 - loss: 0.5618
Epoch 2/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8081 - loss: 0.4894
Epoch 3/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8162 - loss: 0.4633
Epoch 4/10
```

```
750/750 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8138 - loss: 0.4611
Epoch 5/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8151 - loss: 0.4551
Epoch 6/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8147 - loss: 0.4507
Epoch 7/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8215 - loss: 0.4410
Epoch 8/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.8202 - loss: 0.4407
Epoch 9/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.8214 - loss: 0.4393
Epoch 10/10
750/750 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.8247 - loss: 0.4311
<keras.src.callbacks.history.History at 0x7a2af1017d30>
```

```
y_pred = uci_model.predict(X_test)
y_pred_binary = (y_pred > 0.5).astype(int)
```

```
188/188 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```

```
uci_accuracy = accuracy_score(y_test, y_pred_binary)
uci_conf_matrix = confusion_matrix(y_test, y_pred_binary)
uci_class_report = classification_report(y_test, y_pred_binary)
```

```
print(f"Accuracy: {uci_accuracy}")
print("Confusion Matrix:\n", uci_conf_matrix)
print("Classification Report:\n", uci_class_report)
```

```
Accuracy: 0.8156666666666667
Confusion Matrix:
 [[4468  219]
 [ 887  426]]
Classification Report:
               precision    recall  f1-score   support

           0       0.83      0.95      0.89      4687
           1       0.66      0.32      0.44      1313

    accuracy                           0.82      6000
   macro avg       0.75      0.64      0.66      6000
weighted avg       0.80      0.82      0.79      6000
```

```
print("----------ACCURACY SCORE: UCI_Credit_Card.csv----------\n")
print(f'Accuracy score of Logistic Regression is {round(lr_acc_score,4)}')
print(f'Accuracy score of KNeighbors Classifier is {round(knn_acc_score,4)}')
print(f'Accuracy score of DesicionTree Classifier is {round(dt_acc_score,4)}')
print(f'Accuracy score of ANN is {round(uci_accuracy,4)}')
```

```
----------ACCURACY SCORE: UCI_Credit_Card.csv----------

Accuracy score of Logistic Regression is 0.81
Accuracy score of KNeighbors Classifier is 0.7962
Accuracy score of DesicionTree Classifier is 0.7098
Accuracy score of ANN is 0.8157
```

As we can see that the accuracy of Artificial Neural Network (ANN) is slightly higher than other traditional classifiers. So we would use ANN in case we need high accuracy. But we will have to note that it will need more computational power than traditional machine learning models.

Question 2

Regression: Metro Interstate Traffic Volume Dataset :

1. User multi linear regression, polynomial regression and ANN
2. Calculate MSE, R2 score etc.

```
from google.colab import files
uploaded=files.upload()
```

Choose files Metro_Inter..._Volume.csv

- **Metro_Interstate_Traffic_Volume.csv**(text/csv) - 3237208 bytes, last modified: 02/11/2023 - 100% done

```
df=pd.read_csv("Metro_Interstate_Traffic_Volume.csv")
print(df.shape)
print("The first 10 rows of the dataframe")
df.head(10)
```

```
(48204, 9)
The first 5 rows of the dataframe
```

	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	date_time	traffic_volume
0	NaN	288.28	0.0	0.0	40	Clouds	scattered clouds	2012-10-02 09:00:00	5545
1	NaN	289.36	0.0	0.0	75	Clouds	broken clouds	2012-10-02 10:00:00	4516
2	NaN	289.58	0.0	0.0	90	Clouds	overcast clouds	2012-10-02 11:00:00	4767
3	NaN	290.13	0.0	0.0	90	Clouds	overcast clouds	2012-10-02 12:00:00	5026
4	NaN	291.14	0.0	0.0	75	Clouds	broken clouds	2012-10-02 13:00:00	4918
5	NaN	291.72	0.0	0.0	1	Clear	sky is clear	2012-10-02 14:00:00	5181
6	NaN	293.17	0.0	0.0	1	Clear	sky is clear	2012-10-02 15:00:00	5584
7	NaN	293.86	0.0	0.0	1	Clear	sky is clear	2012-10-02 16:00:00	6015
8	NaN	294.14	0.0	0.0	20	Clouds	few clouds	2012-10-02 17:00:00	5791
9	NaN	293.10	0.0	0.0	20	Clouds	few clouds	2012-10-02 18:00:00	4770

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# Data preprocessing
# For multi-linear regression, you can select relevant features
X_multi = df[['temp', 'rain_1h', 'snow_1h', 'clouds_all']]
y_multi = df['traffic_volume']
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_multi, y_multi, test_size=0.2, random_state=42)
```

```
# Multi-linear regression
multi_regression = LinearRegression()
multi_regression.fit(X_train, y_train)
y_pred_multi = multi_regression.predict(X_test)

# Calculate MSE and R2 score for multi-linear regression
mse_multi = mean_squared_error(y_test, y_pred_multi)
r2_multi = r2_score(y_test, y_pred_multi)

# Print the results
print("Multi-linear regression:")
print(f"MSE: {mse_multi}")
print(f"R2 Score: {r2_multi}")
```

```
Multi-linear regression:
MSE: 3860904.796855764
R2 Score: 0.023424420035944693
```

```
# For polynomial regression, you can use PolynomialFeatures
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X_train)
poly_regression = LinearRegression()
poly_regression.fit(X_poly, y_train)

# Transform test data and predict
X_test_poly = poly.transform(X_test)
```