Machine Learning Algorithms to predict the attrition of Bank Customers

Table of Contents

Part One: Introduction and Data Preprocessing

1. Introduction

The banking industry all over the world is characterized by fierce competition among players and the survival of any bank depends largely on its ability to retain customers who are the major reason why the bank is in business. Hence, most product and marketing activities of banks are focused on minimizing customer churn to maximize profits.

In this work, I explored two different supervised machine learning algorithms, K-Nearest Neighbor and Random Forest, to predict customer churn in ABC Multistate bank. The primary objective is to identify the most effective predictive model for customer attrition, which would enable the bank to implement strategies aimed at improving customer retention.

2. Background Research and Literature Review

Analyzing customer attrition is crucial for assessing a bank's performance, and employing data mining techniques, including K-Nearest Neighbors, Decision Trees, Random Forest and Artificial networks, plays a vital role in predicting and assessing customer churn.

Previous research studies carried have underscored the negative impact customer churn on bank revenue and long-term profitability. Yanxuan (2023) identified credit score, age, and active-member status as key factors that determine customer attrition. Sina (2020) study implemented six classification algorithms on an European bank customer dataset and concluded that an Artificial Neural Network structure with a single hidden layer of 5 nodes is the best performing classifier for predicting customer churn.

The Dataset

The dataset used for this work is a publicly available dataset from Kaggle (https://www.kaggle.com/datasets/gauravtopre/bank-customer-churn-dataset/data), and contains information about 10,000 customers of ABC Multistate Bank. The dataset is used for the purpose of determining customer churn, indicating whether a customer has left (1) or remained(0) with the bank. This dataset contains various attributes that was used in predicting attrition for the bank. The details of these features are given in the table below:

S/N	Feature	Definition
1	customer_id	Unique customer ID within the bank
2	credit_score	The credit score of a customer
3	Country	Country of residence of a customer
4	Gender	Denotes where a customer is male or female
5	Age	Age of the customer
6	Tenure	Indicates how long a customer have had an account with the bank
7	Balance	The account balance of a customer
8	products_number	The total number of bank product that a customer has.
9	credit_card	Indicates whether a customer has a credit card or not. '1' means 'yes' and '0' means 'no'
10	active_member	Indicates the customer's active membership status with '1' as active and '0' as not active.
11	estimated_salary	Estimated salary of an account holder.
12	Churn	Represents customer attrition, '1' means 'left' and '0' means 'stayed'.

Features 1 to 11 are the independent variables and feature 12 is the target variable. The table above shows that the dataset does not include personal information like customers' names and addresses, thus avoiding potential ethical concerns related to handling sensitive information.

Explanation and preparation of datasets (Exploratory Data Analysis)

1. The first step taken was to import the necessary libraries required for the analysis, after which the dataset was loaded into a pandas data frame for analysis.

```
# import the required libraries

import numpy as np # To perform a variety of mathematical operations on arrays
import pandas as pd # To analyze, clean, explore, and manipulate data.
import matplotlib.pyplot as plt # For plotting interactive and interesting charts.
import seaborn as sns # An extension of the matplotlib library for more beautiful graphics.
import sklearn # For performing data modeling such as classification and clustering
```

```
# load the dataset

dataset = pd.read_csv('Bank Customer Churn Prediction.csv')
```

2. Using the '.head()' function, the first 5 rows of the dataset were displayed.

```
# checks the first 5 rows of the dataset

dataset.head()
```

	customer_id	credit_score	country	gender	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	churn
0	15634602	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	15647311	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	15619304	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	15701354	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	15737888	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

3. The code below was used to determine the number of rows and columns in the dataset and a brief description of the dataset.

```
# Returns the number of rows and columns

dataset.shape
```

```
(10000, 12)
```

```
# Provides a brief description of the dataset

dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customer_id       10000 non-null  int64  
 1   credit_score      10000 non-null  int64  
 2   country           10000 non-null  object 
 3   gender            10000 non-null  object 
 4   age               10000 non-null  int64  
 5   tenure            10000 non-null  int64  
 6   balance           10000 non-null  float64
 7   products_number   10000 non-null  int64  
 8   credit_card       10000 non-null  int64  
 9   active_member     10000 non-null  int64  
 10  estimated_salary  10000 non-null  float64
 11  churn             10000 non-null  int64  
dtypes: float64(2), int64(8), object(2)
memory usage: 937.6+ KB
```

The output reveals that the dataset has 10,000 rows and 12 columns with no missing values detected across all columns, where the ‘country’ and ‘gender’ columns are categorical variables, and the remaining columns are numeric with ‘integer’ or ‘float’ datatypes.

4. The code below gives a statistical description of the columns of the dataset by using the ‘describe()’ function. It returns the count, mean, standard deviation, minimum, the 1st, and 3rd quartiles, the median and the maximum of each numeric column.

```
# gives a statistical description for the numerical variables in the dataset

dataset.describe(include = 'all')
```

	customer_id	credit_score	country	gender	age	tenure	balance	products_number	credit_card	active_member	estimate
count	1.000000e+04	10000.000000	10000	10000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	1000
unique	NaN	NaN	3	2	NaN	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	France	Male	NaN	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	5014	5457	NaN	NaN	NaN	NaN	NaN	NaN	
mean	1.569094e+07	650.528800	NaN	NaN	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	10009
std	7.193619e+04	96.653299	NaN	NaN	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	5751
min	1.556570e+07	350.000000	NaN	NaN	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	1
25%	1.562853e+07	584.000000	NaN	NaN	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	5100
50%	1.569074e+07	652.000000	NaN	NaN	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	10019
75%	1.575323e+07	718.000000	NaN	NaN	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	14938
max	1.581569e+07	850.000000	NaN	NaN	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	19999

5. The ‘customer_id’ column is irrelevant for the purpose of this work; hence it was dropped to ensure focus on other columns.

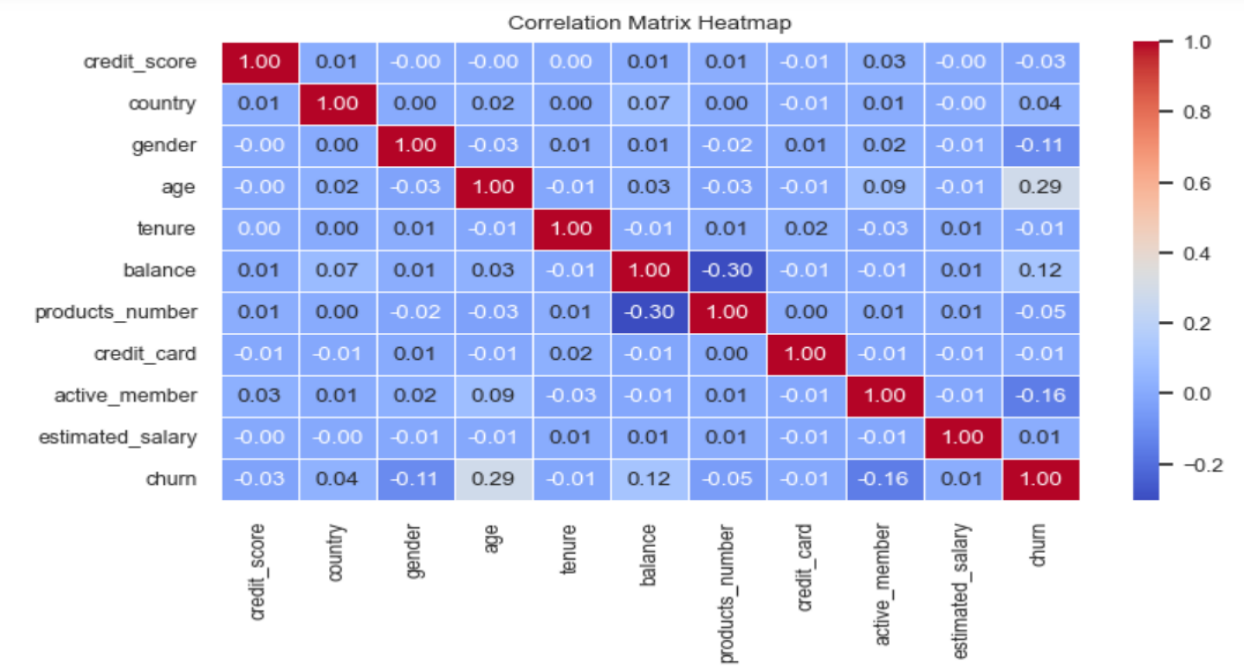
```
# drop 'customer_id' column

dataset.drop(['customer_id'], axis = 1, inplace = True)
```

```
# Confirm that the 'customer_id' column has been dropped

dataset.head()
```

	credit_score	country	gender	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	churn
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

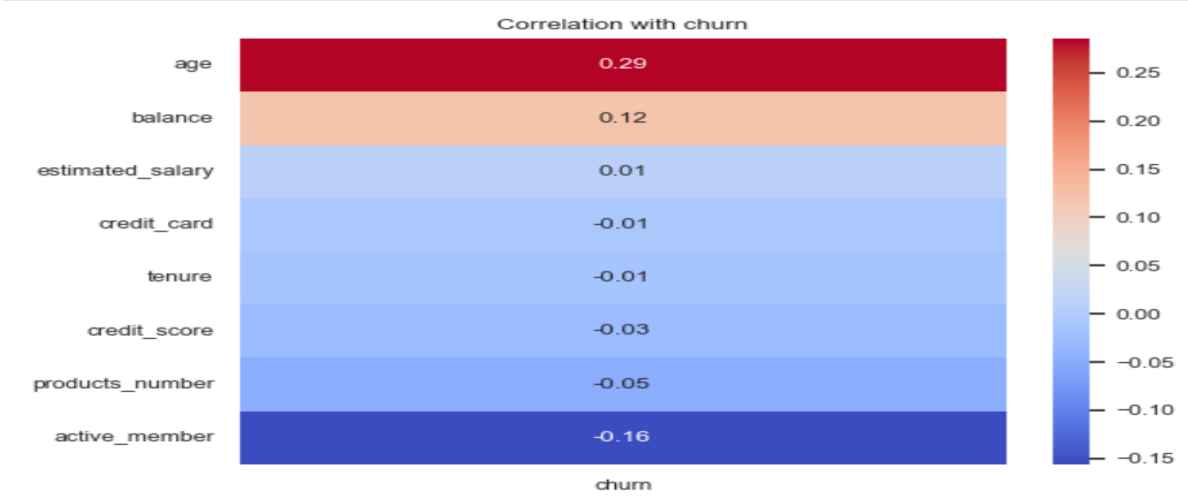
6. In the next step, a correlation matrix was calculated and visualized for the correlation between all variables.

```
# Visualize the correlation of the variables

corr = dataset.corr() # calculates the correlation matrix for the variables

# plots the correlation matrix heat map for all the variables

plt.figure(figsize=(8,4))
sns.heatmap(corr, annot=True, cmap='coolwarm', linewidths=0.5, fmt='.2f')
plt.title("Correlation Matrix Heatmap")
plt.show()
```



7. Since the target variable is ‘churn’, the next lines of code was used to view the correlation of other variables with ‘churn’.

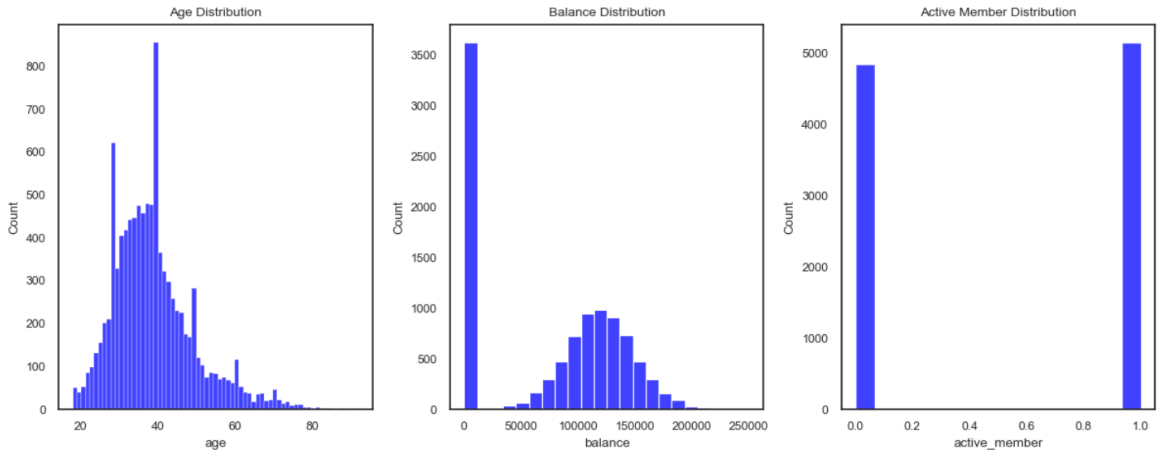
```
# find the correlation of other variables with the target variable 'churn'

def plot_target_correlation(target_variable, dataset):
    # Calculate the correlation with the target variable
    target_corr = corr[target_variable].drop(target_variable)

    # Sort correlation values in descending order
    target_corr_sorted = target_corr.sort_values(ascending=False)

    # Set up the plot
    sns.set(font_scale=0.8)
    sns.set_style("white")
    sns.set_palette("PuBuGn_d")

    # Create a heatmap of the correlations with the target column
    sns.heatmap(target_corr_sorted.to_frame(), cmap="coolwarm", annot=True, fmt='.2f')
    plt.title(f'Correlation with {target_variable}')
    plt.show()

# Display the correlaton of 'churn' with other variables
plot_target_correlation('churn', dataset)
```



The above output showed that the variables with the highest correlation with 'churn' are age, active_member and balance.

8. The distributions of the above highly correlated variables with 'churn' was explored with the code below.

```
# Visualize the distribution of the most correlated numeric features

# Create a figure with three subplots
fig, axes = plt.subplots(1, 3, figsize=(12, 5))

# Plot the distribution of 'age' in the first subplot
sns.histplot(dataset['age'], ax=axes[0], color='blue')
axes[0].set_title('Age Distribution')

# Plot the distribution of 'balance' in the second subplot
sns.histplot(dataset['balance'], ax=axes[1], color='blue')
axes[1].set_title('Balance Distribution')


# Plot the distribution of 'active_member' in the third subplot
sns.histplot(dataset['active_member'], ax=axes[2], color='blue')
axes[2].set_title('Active Member Distribution')

plt.tight_layout()
plt.show()
```



The following can be observed from the above outputs:

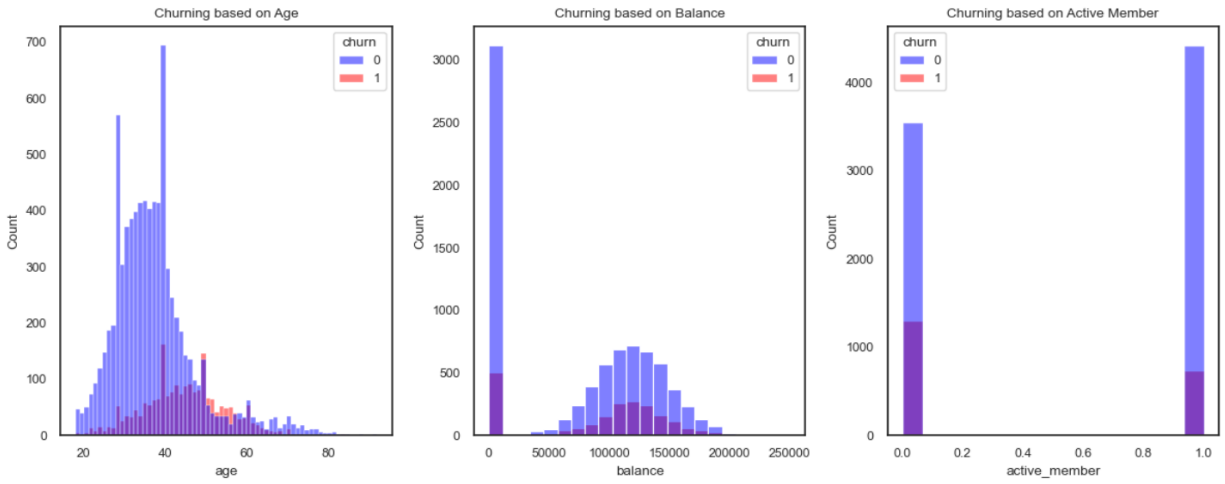
(i) Most of the bank's customers are within the age range of 25 – 50 years.

(ii) A large number of the customers have 'zero' balance.

(iii) The number of active customers of the bank is slightly higher than the number of inactive customers.

9. These features were further visualized by grouping their distributions by 'churn'

```
# Visualize the most correlated numeric features by 'churn'

# Create a figure with three subplots

fig, axes = plt.subplots(1, 3, figsize=(12, 5))

# Explore the age distribution by 'churn'
plt.sca(axes[0])
plt.title("Churning based on Age")
sns.histplot(x="age", hue="churn", data=dataset, palette={0: "blue", 1: "red"})

# Explore the balance distribution by 'churn'
plt.sca(axes[1])
plt.title("Churning based on Balance")
sns.histplot(x="balance", hue="churn", data=dataset, palette={0: "blue", 1: "red"})

# Explore the active_member distribution by 'churn'
plt.sca(axes[2])
plt.title("Churning based on Active Member")
sns.histplot(x="active_member", hue="churn", data=dataset, palette={0: "blue", 1: "red"})

plt.tight_layout()
plt.show()
```



The following findings can be highlighted from the above plots:

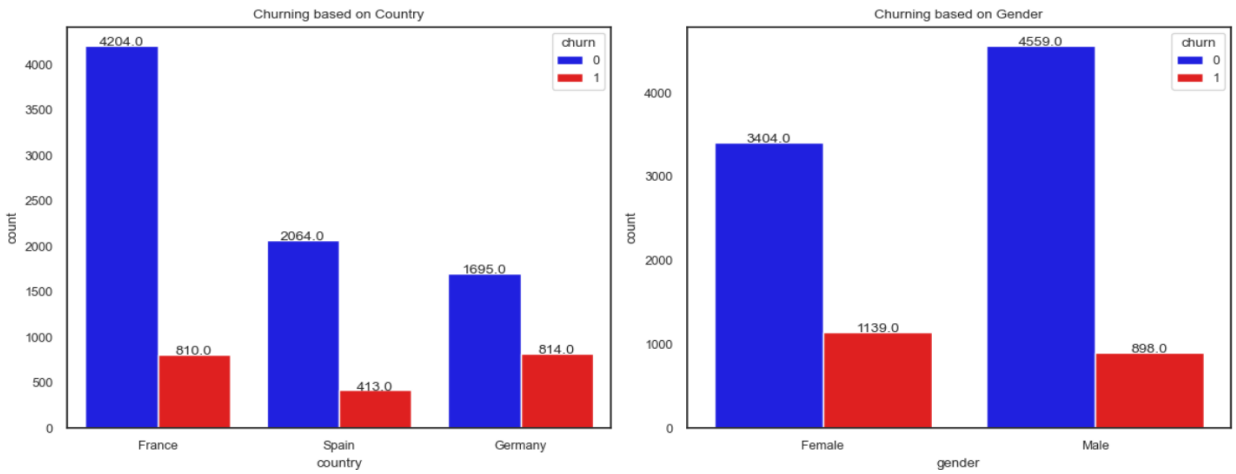
(i) Most of the churners are customers within the age range of 35 – 60 years.

(ii) The highest percentage of the churners have ‘zero’ balance in their accounts.

(iii) Most of the customers that churned are inactive members of the bank.

10. Using the code below, the relationship between the categorical variables and the target variable was visualized.

```
# Explore the categorical features

# Create subplots in a single figure
fig, axes = plt.subplots(1, 2, figsize=(12, 5))

# Explore the country distribution by 'churn'
plt.sca(axes[0])
plt.title("Churning based on Country")
cat_plot = sns.countplot(x="country", hue="churn", data=dataset, palette={0: "blue", 1: "red"})
# Add bar labels
for x in cat_plot.patches:
  cat_plot.text (x.get_x() + x.get_width()  / 2, x.get_height()+ 0.75, x.get_height(),
              horizontalalignment='center', fontsize = 10)

# Explore the gender distribution by 'churn'
plt.sca(axes[1])
plt.title("Churning based on Gender")
cat_plot = sns.countplot(x="gender", hue="churn", data=dataset, palette={0: "blue", 1: "red"})
# Add bar labels
for x in cat_plot.patches:
  cat_plot.text (x.get_x() + x.get_width()  / 2, x.get_height()+ 0.75, x.get_height(),
              horizontalalignment='center', fontsize = 10)

plt.tight_layout()
plt.show()
```



From the above plots;

(i) The country with the highest number of churners is Germany and the country with highest number of customers that remained is France.

(ii) The highest population of the bank's customers are males, but the highest churners are females.

Data Preprocessing

Data preprocessing is a vital step in data mining, ensuring proper data format for model fitting, enhancing success rates, and preventing bias. The following steps outline the dataset preprocessing.

1. Machine learning algorithms require numeric values, hence ‘LabelEncoder()’ was used to convert the categorical columns to numeric values.

```
# Convert categorical variables to numeric using LabelEncoder

from sklearn.preprocessing import LabelEncoder # import the label encoder class

le=LabelEncoder()
dataset["country"]=le.fit_transform(dataset["country"])
dataset["gender"]=le.fit_transform(dataset["gender"])
dataset.head()
```

	credit_score	country	gender	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	churn
0	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	850	2	0	43	2	125510.82	1	1	1	79084.10	0

From the above output, ‘country’ and ‘gender’ columns have been converted to number as follows:

A. Country: - France = 0, Germany = 1, Spain = 2
B. Gender : - Female = 0, Male = 1

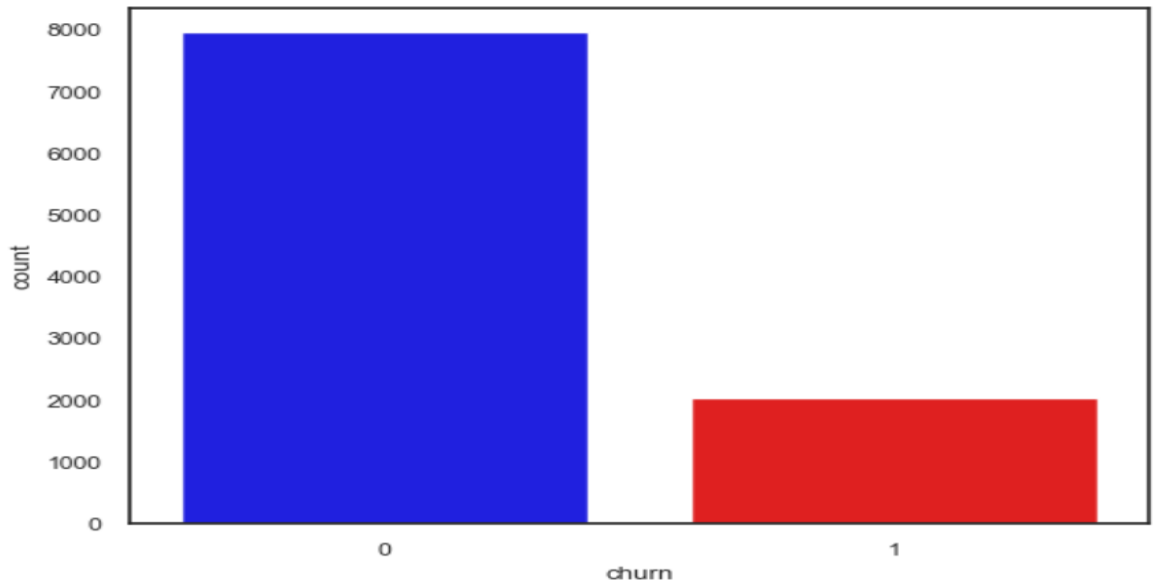
2. Checking the target variable’s class balance is important for fair and accurate algorithm results. The code below was used to verify the balance of the target label’s class.

```
# Check whether the target variable has a balanced class

dataset['churn'].value_counts()
```

```
0    7963
1    2037
Name: churn, dtype: int64
```

```
# Visualize the imbalanced class

sns.countplot(data = dataset, x = 'churn', palette={0: "blue", 1: "red"})
```

```
<Axes: xlabel='churn', ylabel='count'>
```



Based on the above outputs, the target variable has an imbalanced class.

3. The dataset was split into 80% training set and 20% test set, and the class imbalance was addressed using SMOTE; additionally, MinMax scaling was used to normalize the dependent variables to prevent bias from varying features.

```
# split the data into training and test datasets, and handle the class imbalance with SMOTE

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from imblearn.over_sampling import SMOTE

# Define the dependent variable (y) and independent variables(X)
X = dataset.drop('churn', axis = 1)
y = dataset['churn']

# Normalize the 'balance' feature
scaler = MinMaxScaler()
X_sc = scaler.fit_transform(X)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_sc, y, test_size = 0.2, stratify = y, random_state = 0)

# Apply SMOTE for oversampling to handle class imbalance
smote = SMOTE(sampling_strategy='auto', random_state=42)
X_train, y_train = smote.fit_resample(X_train, y_train)
```

The above code defines X as all independent variables, excluding the target variable, while y represents the target variable; ‘random_state’ ensures that the code’s output is reproducible.

4. The code below was used to confirm that the class imbalance issue has been solved.

```
# Confirm that the target variable now has a balanced class in the training data

y_train.value_counts()
```

```
1    6370
0    6370
Name: churn, dtype: int64
```

Part Two: Classification Algorithms

The main objective of this work is to classify bank customers and predict whether a customer will leave the bank or not. To do this, I have applied two different classification algorithms:

1. **K-Nearest Neighbors (KNN):** is a distance-based algorithm that predicts the class of data points by considering the class or average value of its 'k' nearest neighbors.

2. **Random Forest:** is an advanced case of Decision Tree algorithm. It constructs multiple decision trees during training and combines their predictions to make more accurate predictions.

1. Implementation of KNN Algorithm in Python

1. I conducted hyperparameter tuning for the KNN algorithm, varying 'k' values (3, 5, 7, 9, 11) and assessing the model performance with 5-fold cross-validation on the training set to determine the optimal 'k' value for the best model performance.

```python
# Tune hyperparameters and perform a cross-validation on the training dataset

from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier

# Define a range of k values to try
k_values = [3, 5, 7, 9, 11]

# Create empty lists to store the results
mean_scores = []

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)

    # Perform 5-fold cross-validation
    scores = cross_val_score(knn, X_train, y_train, cv=5)

    # Calculate the mean of the scores
    mean_score = scores.mean()
    mean_scores.append(mean_score)

    print(f"Mean Score for k={k}: {mean_score:.2f}")
```

```
Mean Score for k=3: 0.86
Mean Score for k=5: 0.84
Mean Score for k=7: 0.83
Mean Score for k=9: 0.81
Mean Score for k=11: 0.80
```

From the above output, the best value for ‘k’ is 3, which has the highest mean cross validation score.

2. I fitted the KNN classifier to the training set with 'k=3’, using the Minkowski distance metric and p=2 for Euclidean distance, which are the hyperparameters that yielded the highest mean cross-validation score.

```python
# fit the KNN to the training set with the value of k that has the highest mean cv score

from sklearn.neighbors import KNeighborsClassifier

KNN = KNeighborsClassifier(n_neighbors = 3, metric = 'minkowski', p=2)
KNN.fit(X_train, y_train)
```

```
KNeighborsClassifier(n_neighbors=3)
```

3. The code below predicts the class labels for the test dataset using the trained K-nearest neighbors (KNN) classifier and then prints the predicted labels.

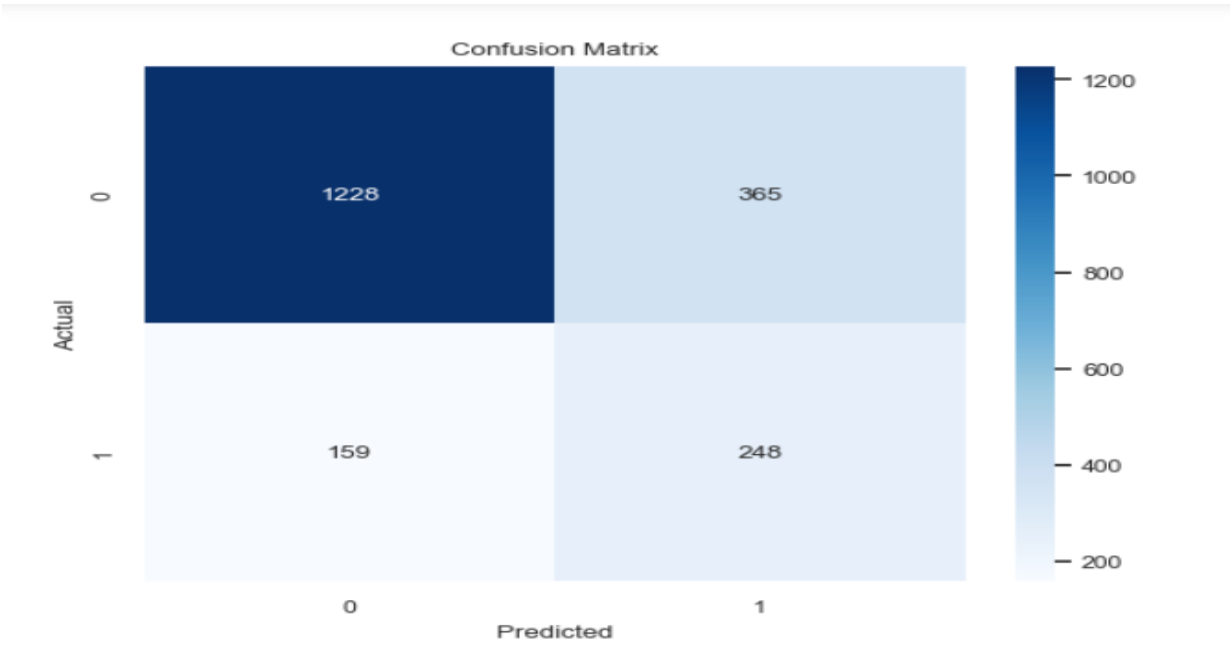
```
# predict the test set result

KNN_y_pred = KNN.predict(X_test)
print(KNN_y_pred)
```

```
[0 0 0 ... 0 1 0]
```

4. The performance of the model was then evaluated by using the accuracy metrics. This was done by the code below which calculates the accuracy, correlation matrix and also generates a classification report for the model.

```
# Evaluate the model performance

from sklearn import metrics # imports metrics

# Calculates the accuracy score for the model
acc = metrics.accuracy_score(y_test, KNN_y_pred)
print('Accuracy:%.2f\n\n'%(acc))

print('-------------------------------------------------')

# Calulates the confusion matrix for the model performance
cm = metrics.confusion_matrix(y_test, KNN_y_pred)
# Create a heatmap of the confusion matrix
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

# Add labels
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')

# Display the plot
plt.show()

print('-------------------------------------------------')

# Generates a classification report for the model
result=metrics.classification_report(y_test,KNN_y_pred)
print('Classification Report:\n')
print(result)
```

```
Accuracy:0.74
```




```
------------------------------------------------
Classification Report:

              precision    recall  f1-score   support

           0       0.89      0.77      0.82      1593
           1       0.40      0.61      0.49       407

    accuracy                           0.74      2000
   macro avg       0.64      0.69      0.66      2000
weighted avg       0.79      0.74      0.76      2000
```

Interpretation of Result:

Based on the above result of the evaluation metrics, the performance of the KNN model can be interpreted as follows:

- **Accuracy:** measures the ratio of correct predictions by the model on the test dataset, the above result showed that the model was able to predict 74% of the test dataset correctly.
- **Confusion Matrix:** the model predicted 1228 customers that stayed (TP) and 248 customers that left (TN) correctly, while it predicted 365 customers that did not leave as ‘churn’ (FN) and 159 customers that left as ‘stayed’ (FP).
- **Precision:** measures the accuracy of true predictions by the model. For class ‘0’, a precision of 0.89 implies that the model is 80% correct whenever it predicts that a customer will stay while for class ‘1’, it has a precision score of 0.40, which implies that the model is only 40% accurate when it predicts that a customer will churn.

- **Recall:** measures the model's ability to correctly identify all the customers that fall within a particular class. The model correctly identifies 77% of customers that stayed and identifies 61% of those that left.

2. Implementation of Random Forest in Python

1. The code below was used to perform hyperparameter tuning and Grid Search cross-validation for the Random Forest Classifier. This was done to find the best combination of parameters that would give the best performance of the model. The code also prints the best parameters and corresponding mean cross-validation scores with standard deviations.

```
# Hyperparameter tuning and cross validation(Grid Search)

# Define a function to print the best parameters after cross-validation
def print_results(results):
    print('BEST PARAMS: {}\n'.format(results.best_params_))

    means = results.cv_results_['mean_test_score']
    stds = results.cv_results_['std_test_score']
    for mean, std, params in zip(means, stds, results.cv_results_['params']):
        print('{} (+/-{}) for {}'.format(round(mean, 3), round(std * 2, 3), params))

# Import the Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(random_state = 42)
parameters = {
    'n_estimators': [5, 50, 100],
    'max_depth': [2, 10, 20, None]
}

# Import Grid Search cross-validation
from sklearn.model_selection import GridSearchCV

# Perform 5-fold cross-validation
cv = GridSearchCV(rf, parameters, cv=5)
cv.fit(X_train, y_train.values.ravel())

print_results(cv)
```

```
BEST PARAMS: {'max_depth': 20, 'n_estimators': 100}

0.727 (+/-0.024) for {'max_depth': 2, 'n_estimators': 5}
0.77 (+/-0.04) for {'max_depth': 2, 'n_estimators': 50}
0.774 (+/-0.032) for {'max_depth': 2, 'n_estimators': 100}
0.841 (+/-0.062) for {'max_depth': 10, 'n_estimators': 5}
0.856 (+/-0.069) for {'max_depth': 10, 'n_estimators': 50}
0.856 (+/-0.069) for {'max_depth': 10, 'n_estimators': 100}
0.868 (+/-0.059) for {'max_depth': 20, 'n_estimators': 5}
0.898 (+/-0.065) for {'max_depth': 20, 'n_estimators': 50}
0.899 (+/-0.07) for {'max_depth': 20, 'n_estimators': 100}
0.867 (+/-0.058) for {'max_depth': None, 'n_estimators': 5}
0.897 (+/-0.069) for {'max_depth': None, 'n_estimators': 50}
0.899 (+/-0.07) for {'max_depth': None, 'n_estimators': 100}
```

From the above output, the model will give its best performance when 'max_depth = 20' and 'n_estimators = 100'.

2. The next code fits the best parameters on the training dataset.

```
# fit the model with the best parameters on the dataset

rf = RandomForestClassifier(n_estimators=100, max_depth=20)
rf.fit(X_train, y_train.values.ravel())
```

```
RandomForestClassifier(max_depth=20)
```

3. The code below predicts the class labels for the test dataset using the trained Random Forest classifier and then prints the predicted labels.

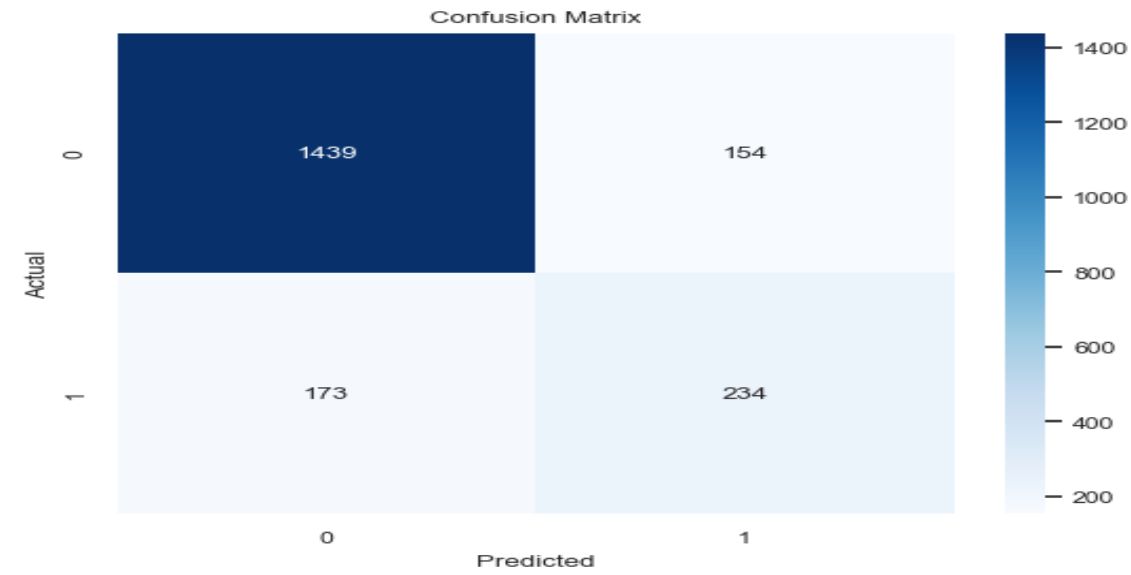
```
# predict the test set result

rf_y_pred = rf.predict(X_test)
print(rf_y_pred)
```

```
[0 0 0 ... 0 1 0]
```

4. The performance of the model was then evaluated by using the accuracy metrics. This was done by the code below which calculates the accuracy, correlation matrix and also generates a classification report for the model.

```
# Evaluate the random forest model performance

# Calculates the accuracy score for the model
acc = metrics.accuracy_score(y_test, rf_y_pred)
print('Accuracy:%.2f\n\n'%(acc))

print('-------------------------------------------------')

# Calulates the confusion matrix for the model performance
cm = metrics.confusion_matrix(y_test, rf_y_pred)
# Create a heatmap of the confusion matrix
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

# Add labels
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')

# Display the plot
plt.show()

print('-------------------------------------------------')

# Generates a classification report for the model
result=metrics.classification_report(y_test,rf_y_pred)
print('Classification Report:\n')
print(result)
```

```
Accuracy:0.84
```




```
------------------------------------------------
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.90      0.90      1593
           1       0.60      0.57      0.59       407

    accuracy                           0.84      2000
   macro avg       0.75      0.74      0.74      2000
weighted avg       0.83      0.84      0.84      2000
```

Interpretation of Result:

From the above result of the evaluation metrics, the performance of the Random Forest classifier was interpreted as follows:

- **Accuracy:** The model achieved an accuracy score of 0.84, which implies that 84% of the test dataset was predicted correctly by the model.
- **Confusion Matrix:** The model correctly predicted 1439 cases of customers who stayed (TP) and 234 cases of customers who churned (TN), while it inaccurately predicted 154 customers as left when they did not (FN) and missed 173 customers who actually left the bank (FP).
- **Precision:** the model has a precision score of 0.89 and 0.60 for class ‘0’ and class ‘1’ respectively. This implies that the model is 89% correct when it predicts customers that stayed and 60% accurate when it predicts customers that churned.
- **Recall:** the random forest model correctly identifies 90% of customers that stayed and 57% of those that left.

3. Results and Discussion

Table of results of the two classification algorithms

Evaluation Metrics	K-NEAREST NEIGHBORS	RANDOM FOREST
Accuracy	0.74	0.84
Precision	**0:** 0.89 **1:** 0.40	**0:** 0.89 **1:** 0.60
Recall	**0:** 0.77 **1:** 0.61	**0:** 0.90 **1:** 0.57

From the above table of results, it can be observed that Random Forest classifier achieved a higher accuracy score of 84%, indicating that it has good overall performance in predicting customer churns. While KNN had a relatively low precision score of 40% for churn, Random Forest had a precision score of 60%.

Based on the comparison of the two classifiers, Random Forest outperformed KNN on all metrics. It achieves a better balance between precision and recall for both churn and non-churn classes, making it a more effective algorithm for predicting customer churn based on the data.

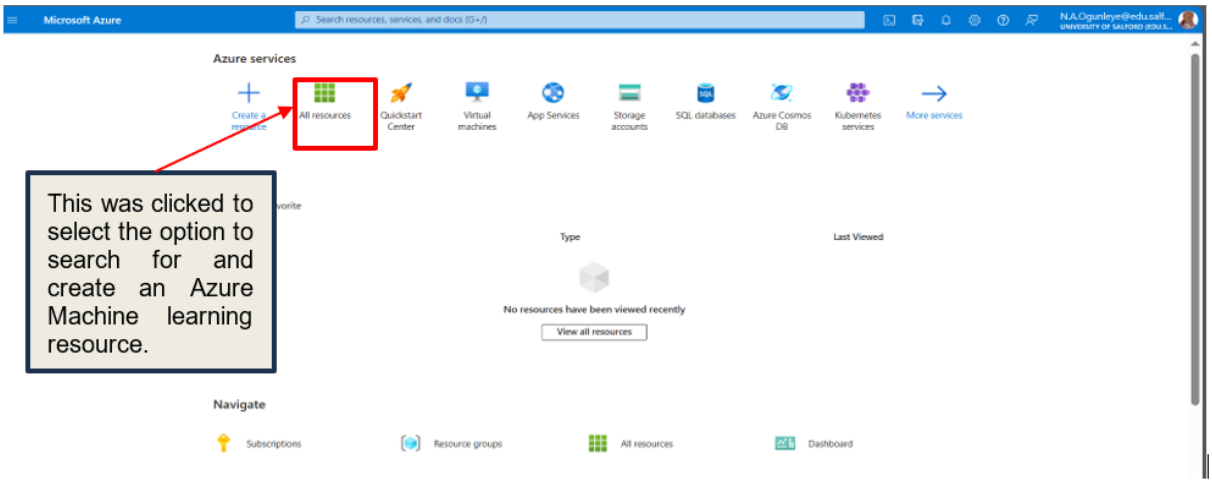
4. Implementation in Azure Machine Learning Designer

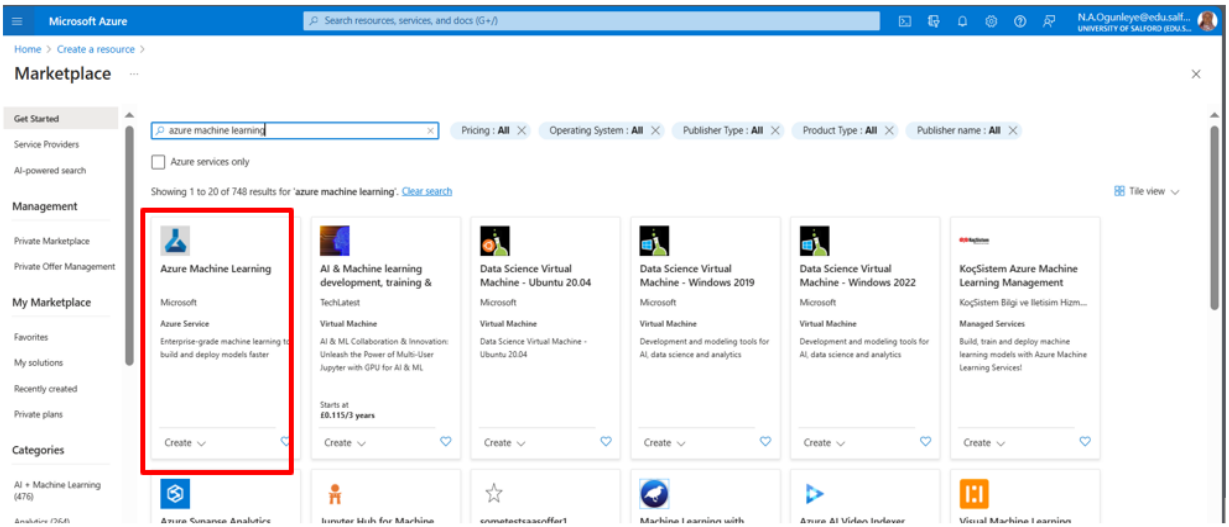
In this section, I used Azure Machine Learning designer to apply two classification algorithms on the Bank Customer Churn dataset:

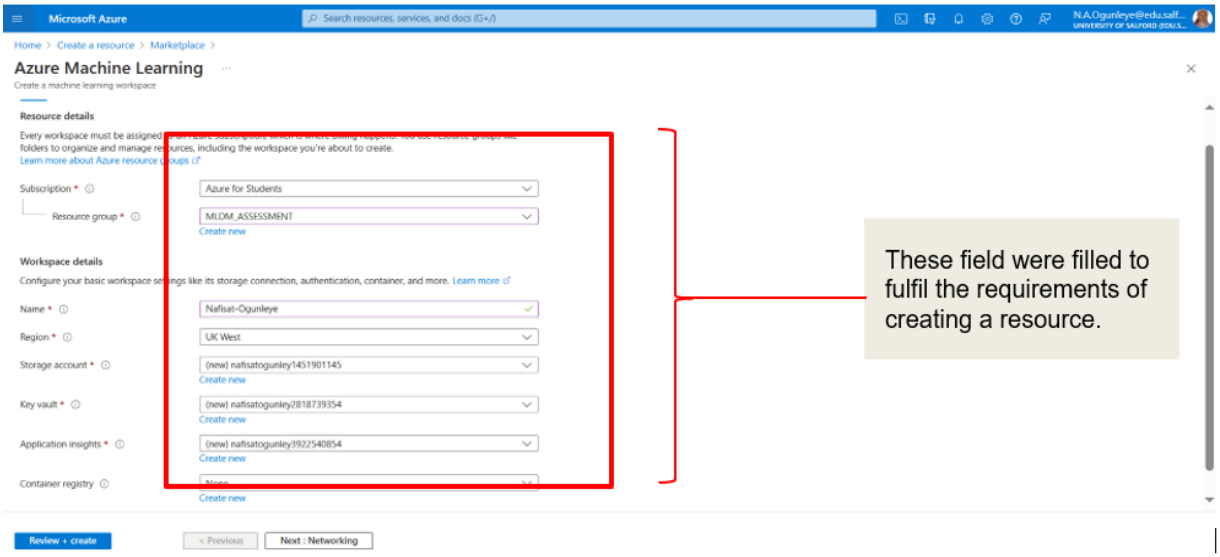
(a) **Two-Class Boosted Decision Tree:** is an ensemble model for binary classification which builds decision trees to improve overall predictive performance.

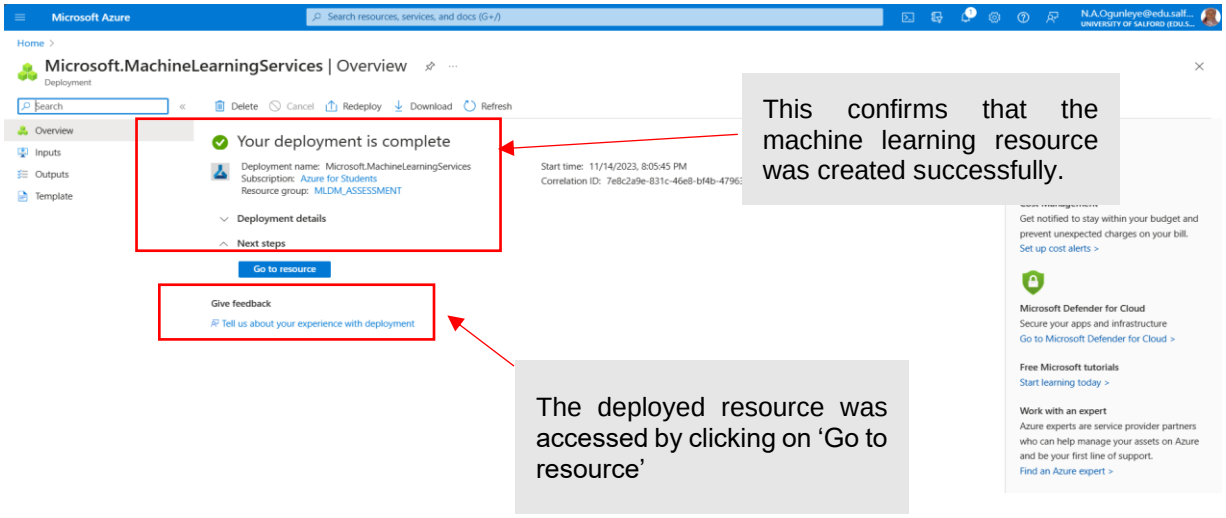
(b) **Two-Class Neural Network:** is an artificial neural network for binary classification that uses interconnected nodes in layers to identify patterns in data by adjusting weights during training.

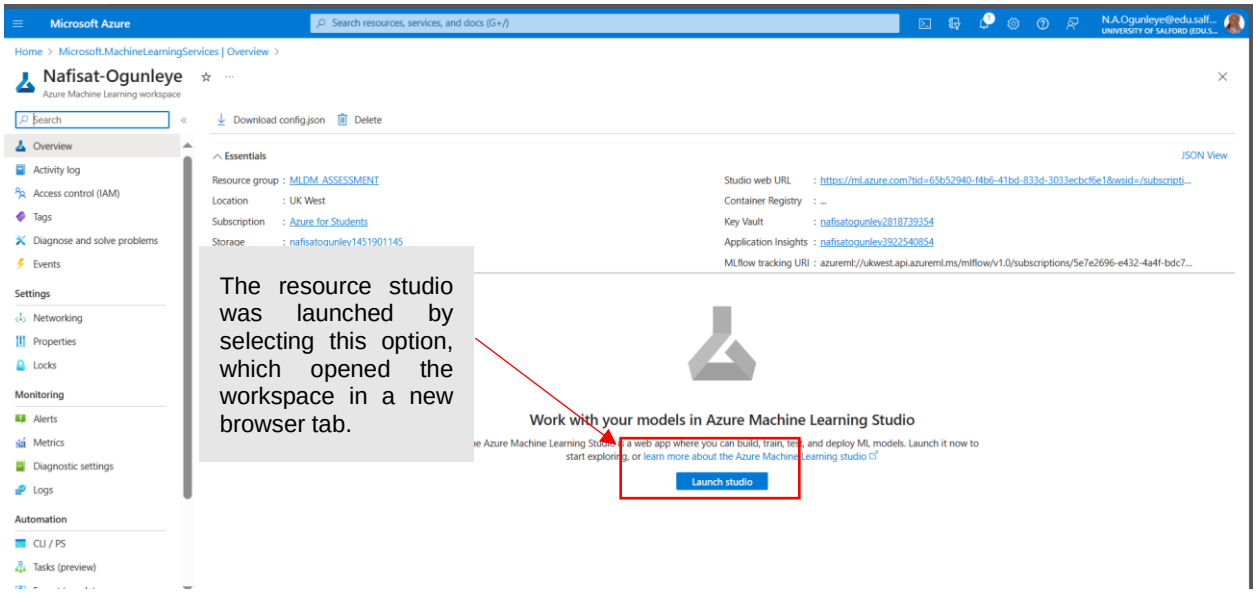
1. I started by signing into my Azure student account portal and creating an Azure machine learning resource. This was carried out by executing the following steps:


Microsoft Azure
Azure services
Create a resource
All resources
Quickstart Center
Virtual machines
App Services
Storage accounts
SQL databases
Azure Cosmos DB
Kubernetes services
More services
This was clicked to select the option to search for and create an Azure Machine learning resource.
Type
Last Viewed
No resources have been viewed recently
View all resources
Navigate
Subscriptions
Resource groups
All resources
Dashboard

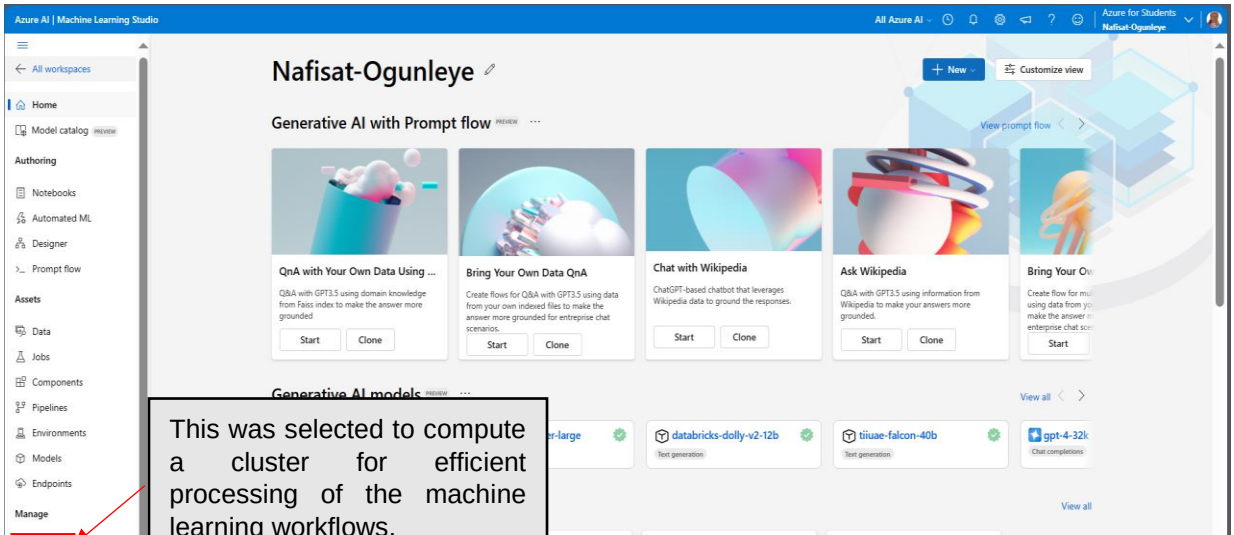

Microsoft Azure
Home > Create a resource >
Marketplace
azure machine learning
Pricing : All
Operating System : All
Publisher Type : All
Product Type : All
Publisher name : All
Azure services only
Showing 1 to 20 of 748 results for 'azure machine learning'. Clear search
Tile view
Azure Machine Learning
AI & Machine learning development, training &
Data Science Virtual Machine - Ubuntu 20.04
Data Science Virtual Machine - Windows 2019
Data Science Virtual Machine - Windows 2022
KoçSistem Azure Machine Learning Management
Create

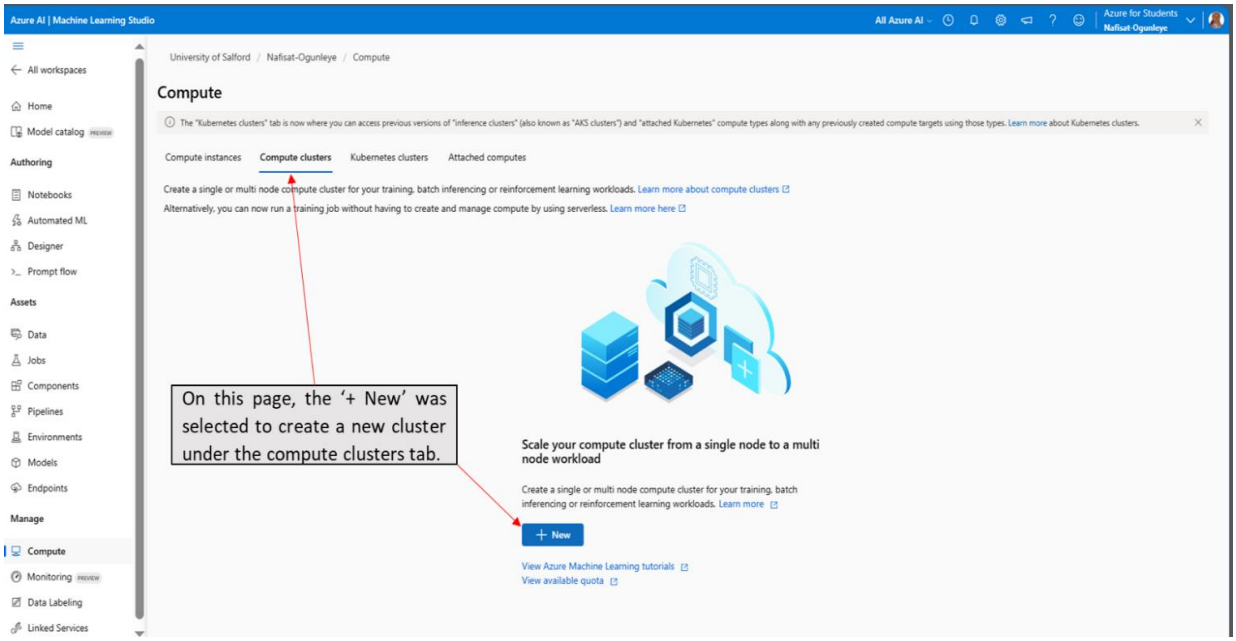

Microsoft Azure
Home > Create a resource > Marketplace >
Azure Machine Learning
Create a machine learning workspace
Resource details
Subscription *
Azure for Students
Resource group *
MLOM_ASSESSMENT
Create new
Workspace details
Name *
Nafisat-Ogunleye
Region *
UK West
Storage account *
(new) nafisatogunley1451901145
Key vault *
(new) nafisatogunley2818739354
Application insights *
(new) nafisatogunley3922540854
Container registry
None
Review + create
< Previous
Next : Networking
These field were filled to fulfil the requirements of creating a resource.

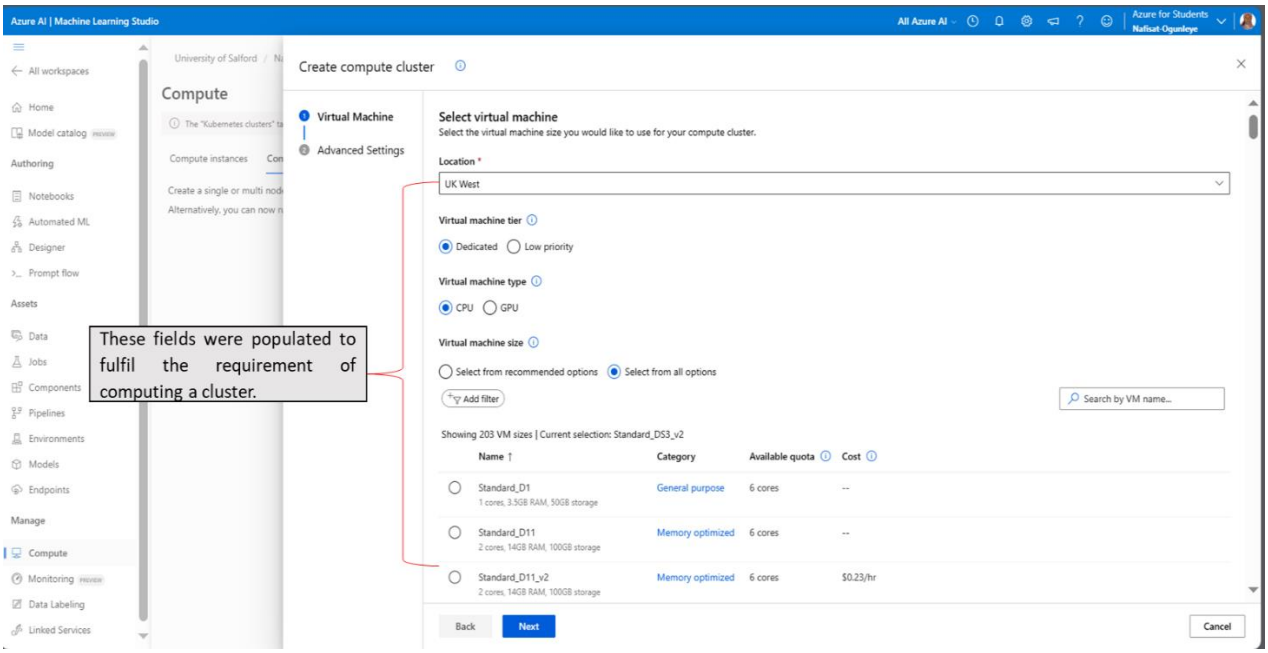


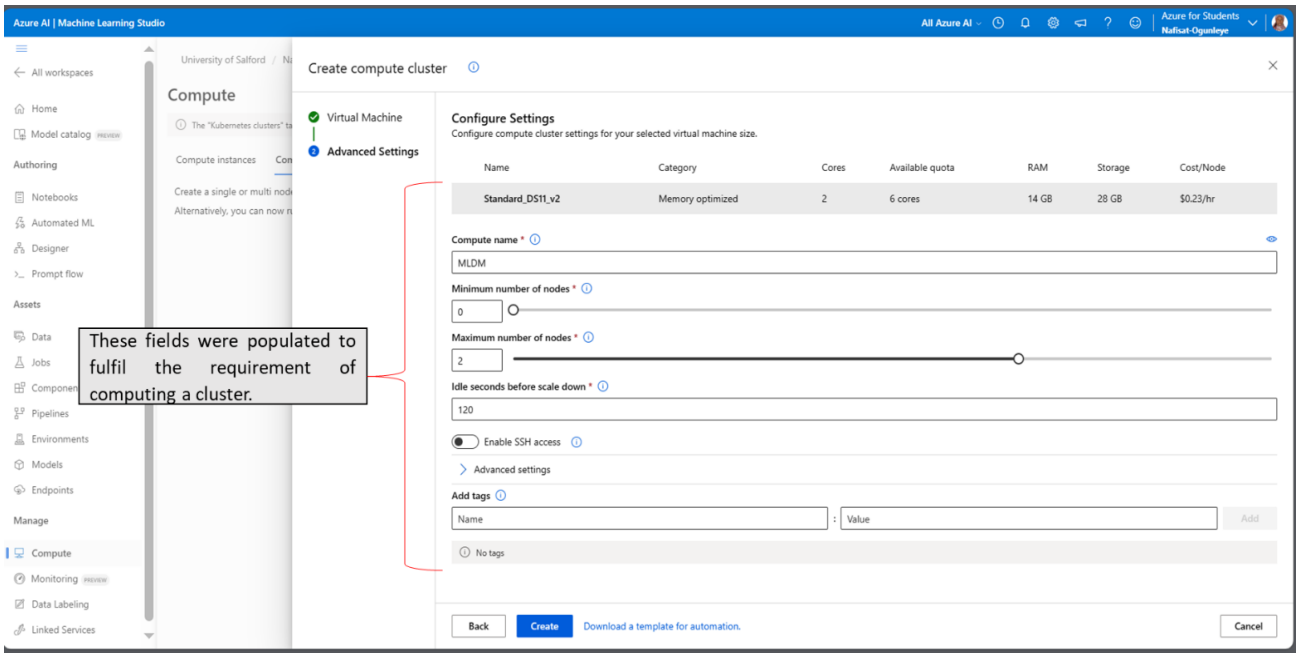


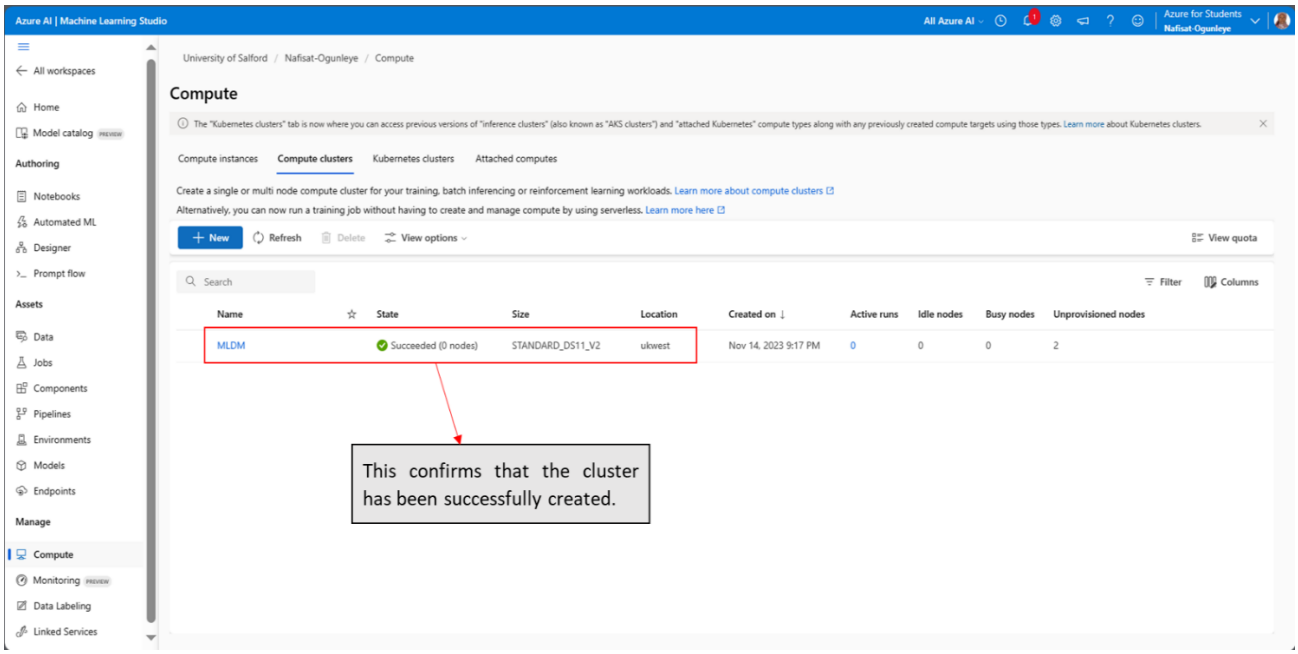
2. The below image shows the interface of the launched machine learning resource studio workspace whose resources can be managed by using the left-hand pane in the workspace. The next steps shows the creating of a cluster for the workspace.



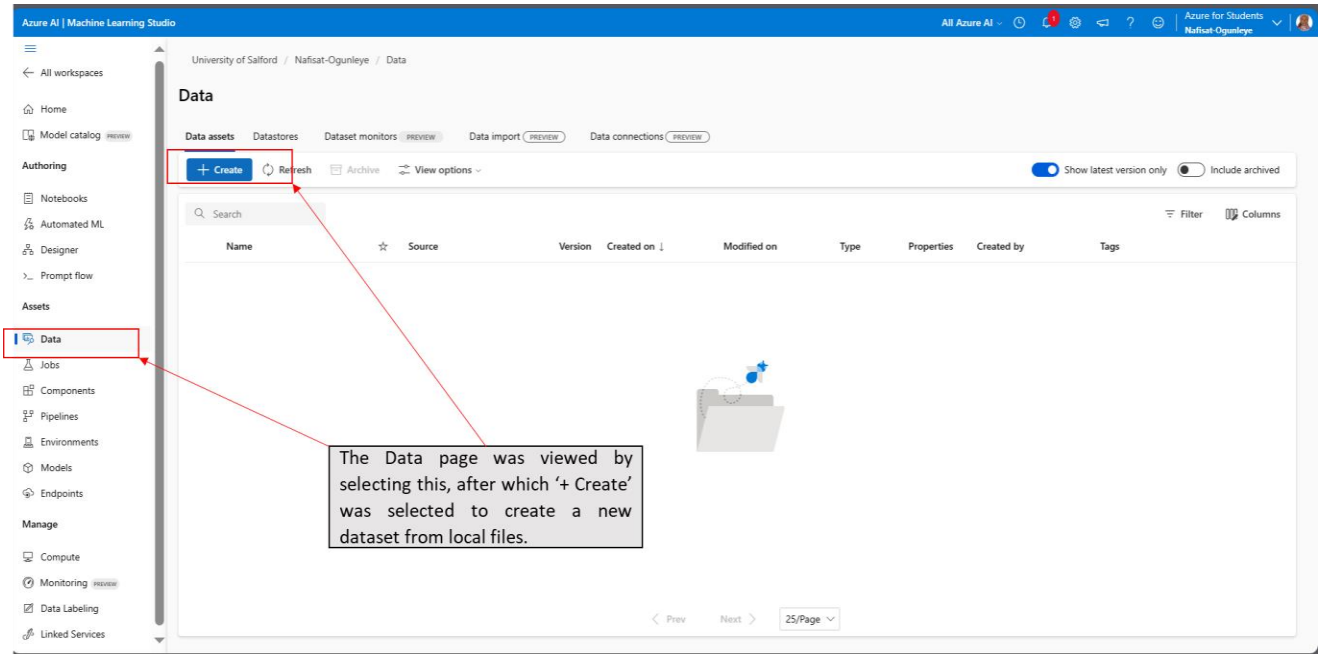

Azure AI | Machine Learning Studio
All Azure AI
Azure for Students
Nafisat-Ogunleye
All workspaces
Home
Model catalog
Authoring
Notebooks
Automated ML
Designer
Prompt flow
Assets
Data
Jobs
Components
Pipelines
Environments
Models
Endpoints
Manage
Compute
Monitoring
Data Labeling
Linked Services
University of Salford / Nafisat-Ogunleye / Compute
Compute
The "Kubernetes clusters" tab is now where you can access previous versions of "inference clusters" (also known as "AKS clusters") and "attached Kubernetes" compute types along with any previously created compute targets using those types. Learn more about Kubernetes clusters.
Compute instances
Compute clusters
Kubernetes clusters
Attached computes
Create a single or multi node compute cluster for your training, batch inferencing or reinforcement learning workloads. Learn more about compute clusters
Alternatively, you can now run a training job without having to create and manage compute by using serverless. Learn more here
On this page, the '+ New' was selected to create a new cluster under the compute clusters tab.
Scale your compute cluster from a single node to a multi node workload
Create a single or multi node compute cluster for your training, batch inferencing or reinforcement learning workloads. Learn more
+ New
View Azure Machine Learning tutorials
View available quota

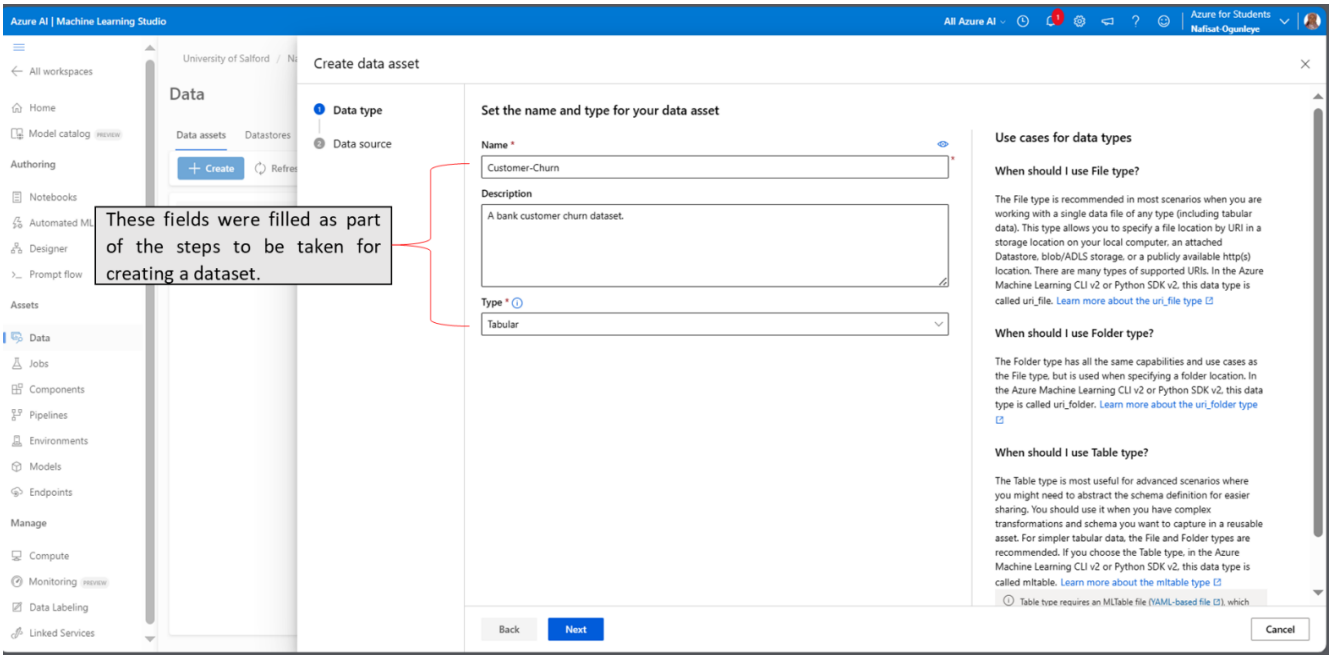

Azure AI | Machine Learning Studio
All Azure AI
Azure for Students
Nafisat-Ogunleye
Create compute cluster
Virtual Machine
Advanced Settings
Select virtual machine
Select the virtual machine size you would like to use for your compute cluster.
Location *
UK West
Virtual machine tier
Dedicated
Low priority
Virtual machine type
CPU
GPU
Virtual machine size
Select from recommended options
Select from all options
Add filter
Search by VM name...
Showing 203 VM sizes | Current selection: Standard_DS3_v2
Name
Category
Available quota
Cost
Standard_D1
1 cores, 3.5GB RAM, 50GB storage
General purpose
6 cores
--
Standard_D11
2 cores, 14GB RAM, 100GB storage
Memory optimized
6 cores
--
Standard_D11_v2
2 cores, 14GB RAM, 100GB storage
Memory optimized
6 cores
$0.23/hr
These fields were populated to fulfil the requirement of computing a cluster.
Back
Next
Cancel

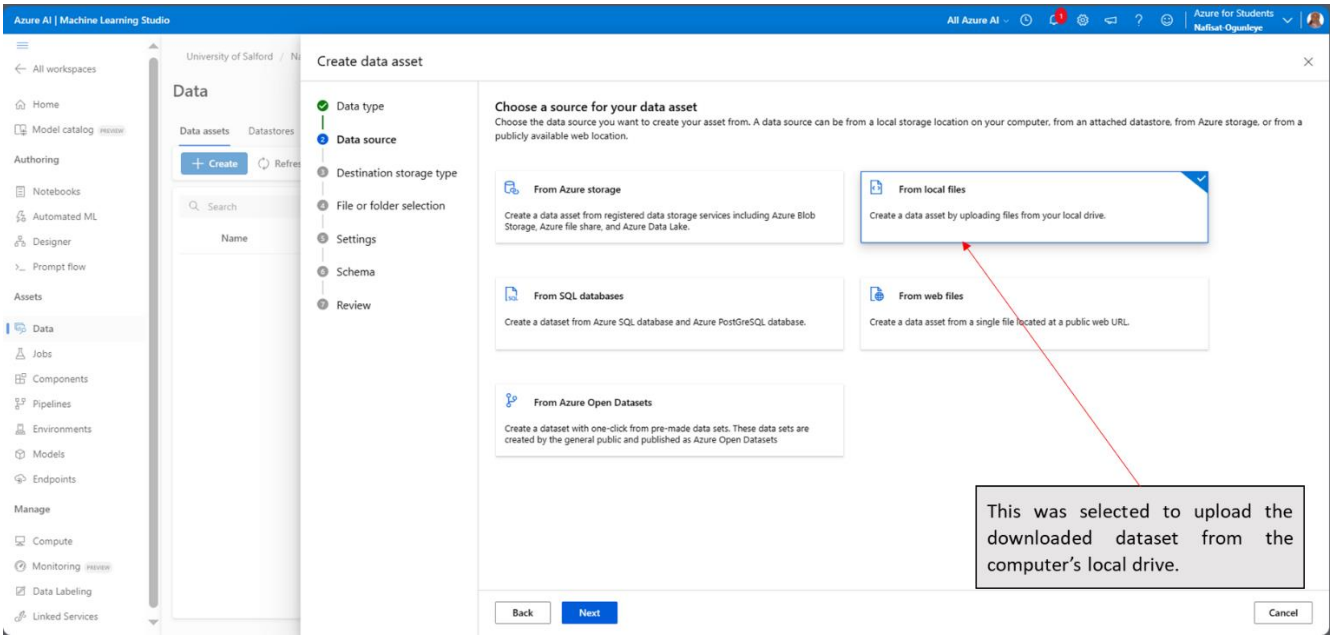


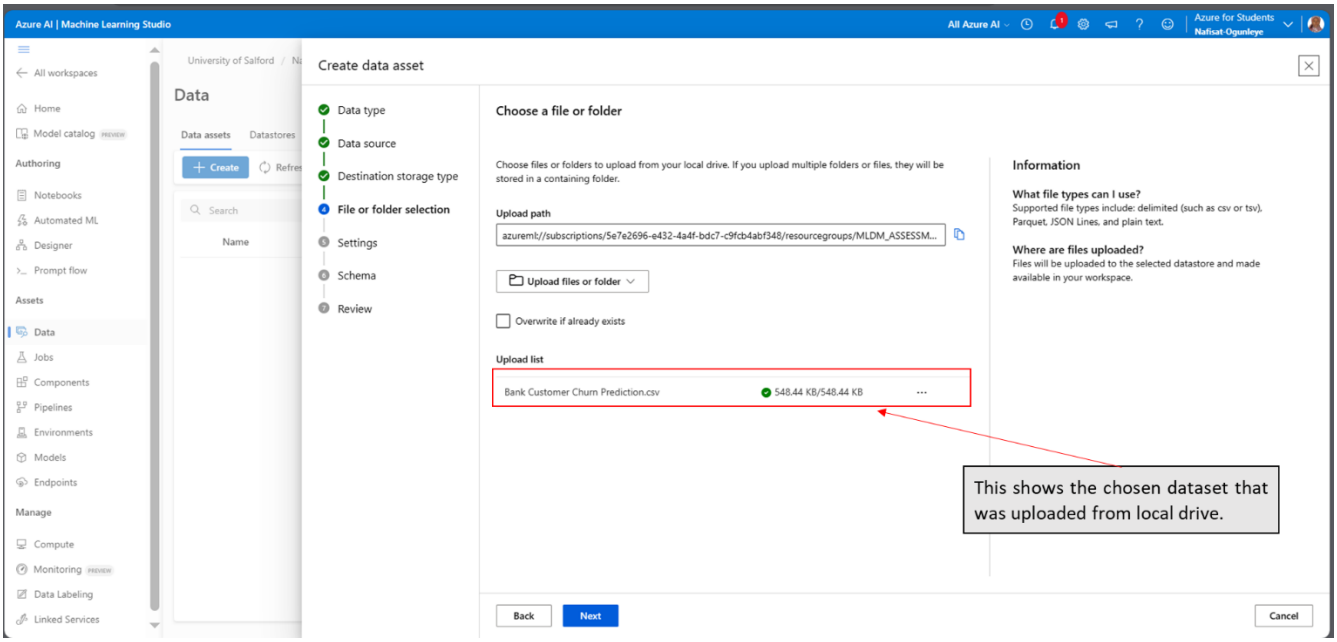


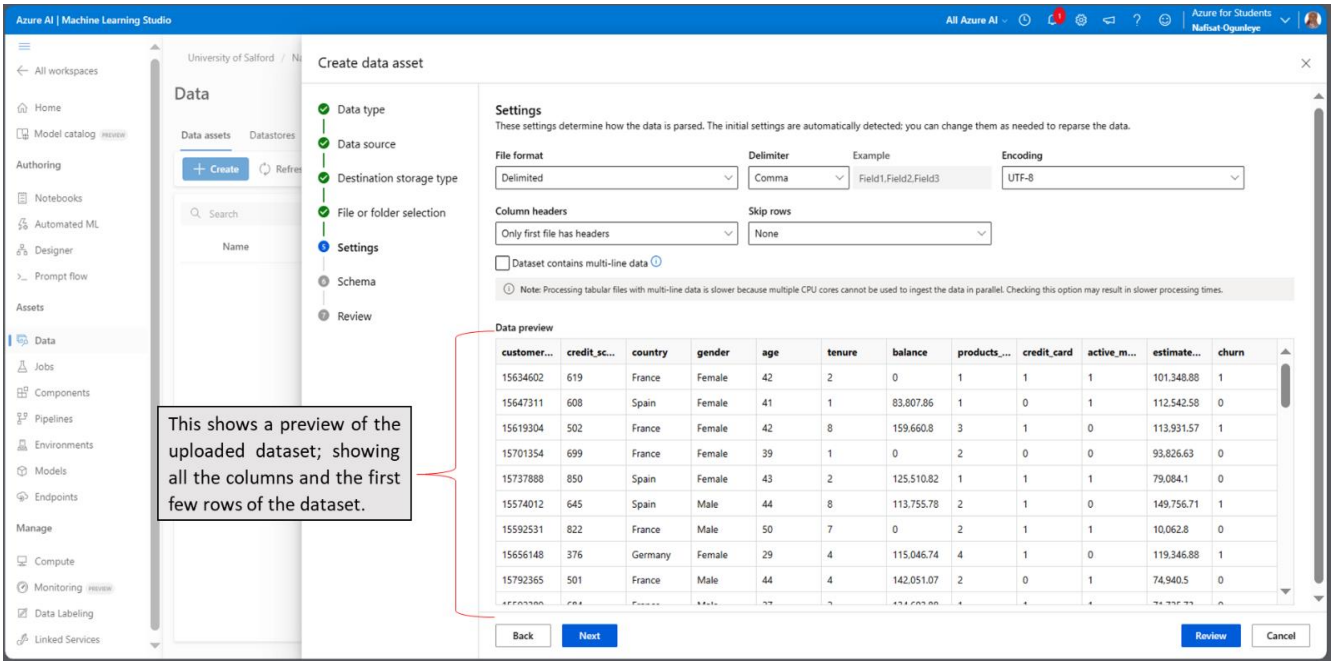
3. In the next step, the dataset that will be used for analysis was imported into the workspace using the steps below.

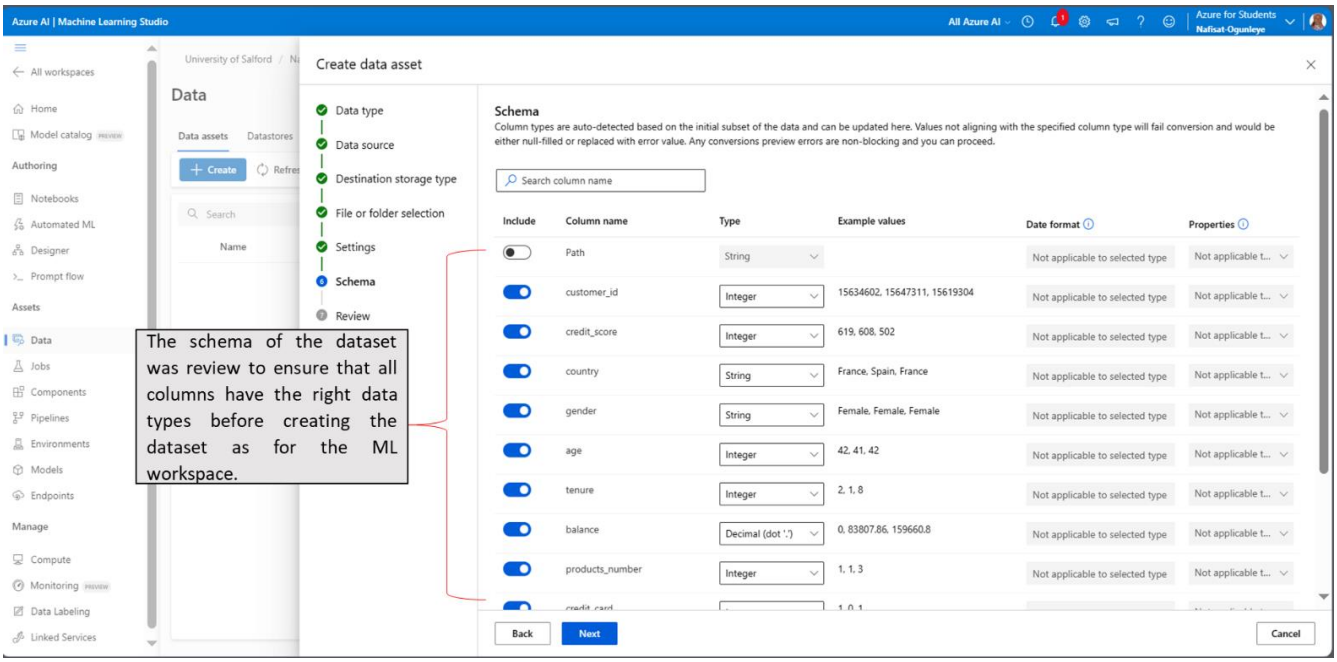

Azure AI | Machine Learning Studio
All Azure AI
Azure for Students
Nafisat-Ogunleye
All workspaces
Home
Model catalog
Authoring
Notebooks
Automated ML
Designer
Prompt flow
Assets
Data
Jobs
Components
Pipelines
Environments
Models
Endpoints
Manage
Compute
Monitoring
Data Labeling
Linked Services
University of Salford / Nafisat-Ogunleye / Data
Data
Data assets
Datastores
Dataset monitors
Data import
Data connections
Create
Refresh
Archive
View options
Show latest version only
Include archived
Search
Filter
Columns
Name
Source
Version
Created on
Modified on
Type
Properties
Created by
Tags
Prev
Next
25/Page
The Data page was viewed by selecting this, after which '+ Create' was selected to create a new dataset from local files.


Azure AI | Machine Learning Studio
All Azure AI
Azure for Students
Nafisat-Ogunleye
Create data asset
Data type
Data source
Set the name and type for your data asset
Name *
Customer-Churn
Description
A bank customer churn dataset.
Type *
Tabular
These fields were filled as part of the steps to be taken for creating a dataset.
Use cases for data types
When should I use File type?
The File type is recommended in most scenarios when you are working with a single data file of any type (including tabular data). This type allows you to specify a file location by URI in a storage location on your local computer, an attached Datastore, blob/ADLS storage, or a publicly available http(s) location. There are many types of supported URIs. In the Azure Machine Learning CLI v2 or Python SDK v2, this data type is called uri_file. Learn more about the uri_file type
When should I use Folder type?
The Folder type has all the same capabilities and use cases as the File type, but is used when specifying a folder location. In the Azure Machine Learning CLI v2 or Python SDK v2, this data type is called uri_folder. Learn more about the uri_folder type
When should I use Table type?
The Table type is most useful for advanced scenarios where you might need to abstract the schema definition for easier sharing. You should use it when you have complex transformations and schema you want to capture in a reusable asset. For simpler tabular data, the File and Folder types are recommended. If you choose the Table type, in the Azure Machine Learning CLI v2 or Python SDK v2, this data type is called mltable. Learn more about the mltable type
Back
Next
Cancel

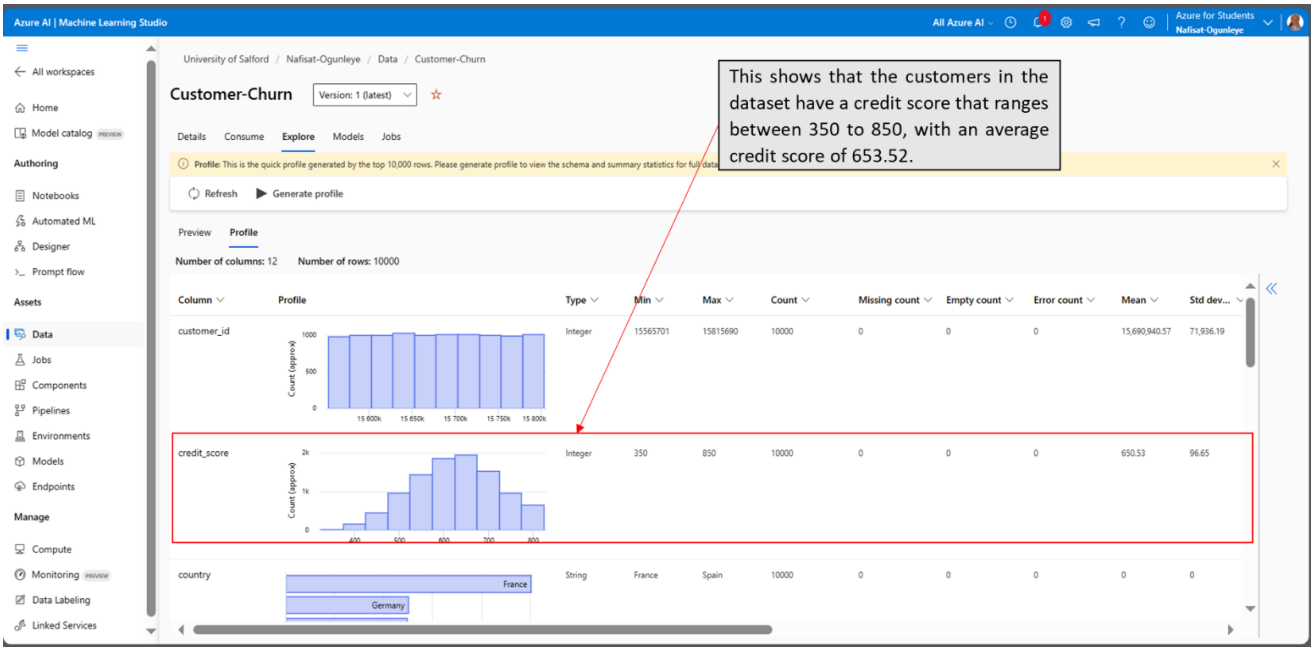

Azure AI | Machine Learning Studio
Create data asset
Data type
Data source
Destination storage type
File or folder selection
Settings
Schema
Review
Choose a source for your data asset
Choose the data source you want to create your asset from. A data source can be from a local storage location on your computer, from an attached datastore, from Azure storage, or from a publicly available web location.
From Azure storage
Create a data asset from registered data storage services including Azure Blob Storage, Azure file share, and Azure Data Lake.
From local files
Create a data asset by uploading files from your local drive.
From SQL databases
Create a dataset from Azure SQL database and Azure PostGreSQL database.
From web files
Create a data asset from a single file located at a public web URL.
From Azure Open Datasets
Create a dataset with one-click from pre-made data sets. These data sets are created by the general public and published as Azure Open Datasets
This was selected to upload the downloaded dataset from the computer's local drive.
Back
Next
Cancel

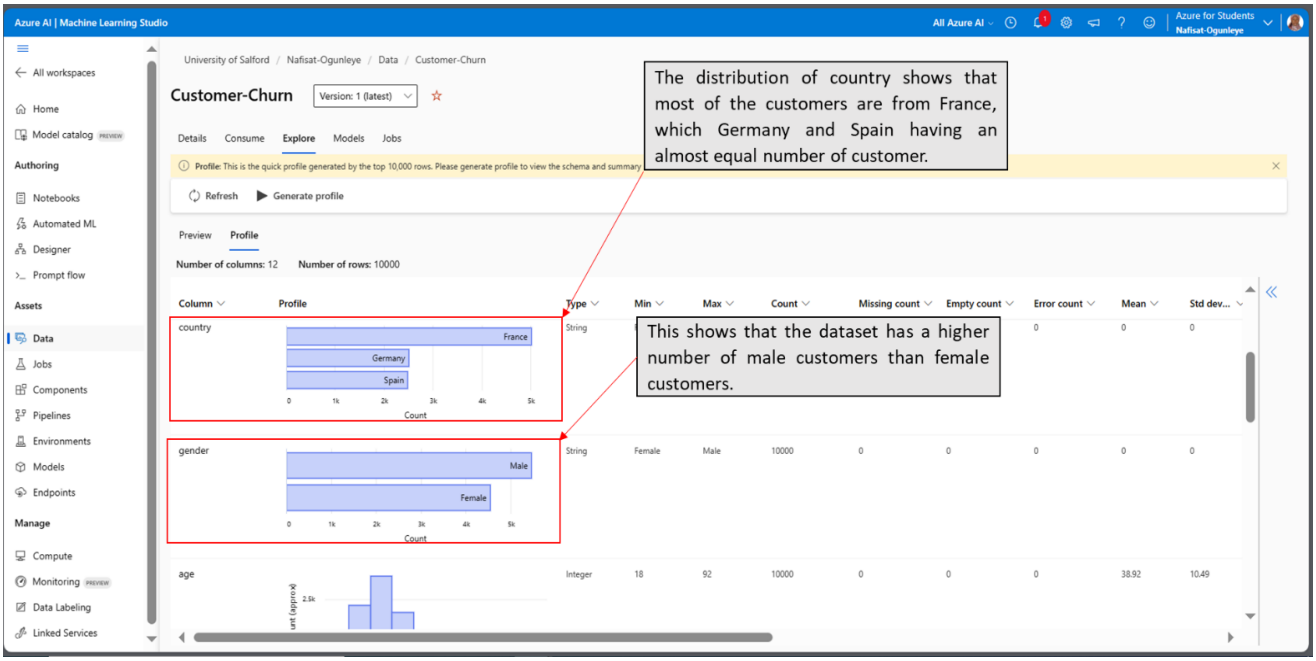

Azure AI | Machine Learning Studio
Create data asset
Data type
Data source
Destination storage type
File or folder selection
Settings
Schema
Review
Choose a file or folder
Choose files or folders to upload from your local drive. If you upload multiple folders or files, they will be stored in a containing folder.
Upload path
azureml://subscriptions/5e7e2696-e432-4a4f-bdc7-c9fcb4abf348/resourcegroups/MLDM_ASSESSM...
Upload files or folder
Overwrite if already exists
Upload list
Bank Customer Churn Prediction.csv
548.44 KB/548.44 KB
Information
What file types can I use?
Supported file types include: delimited (such as csv or tsv), Parquet, JSON Lines, and plain text.
Where are files uploaded?
Files will be uploaded to the selected datastore and made available in your workspace.
This shows the chosen dataset that was uploaded from local drive.
Back
Next
Cancel

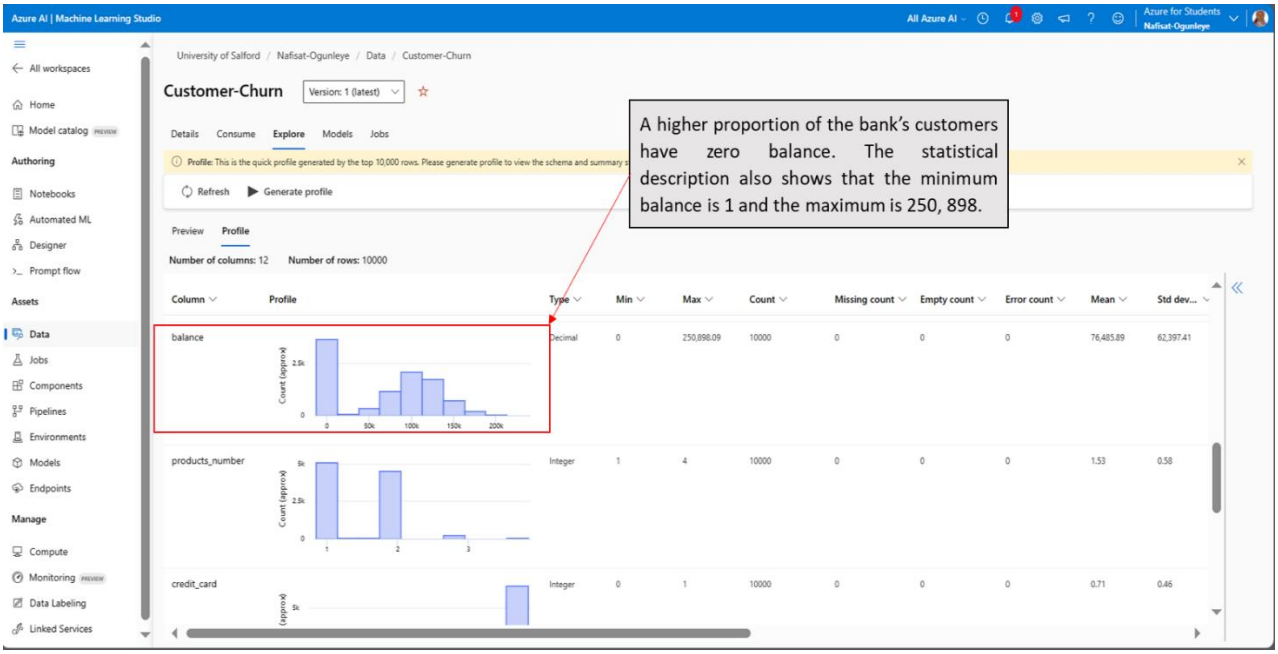


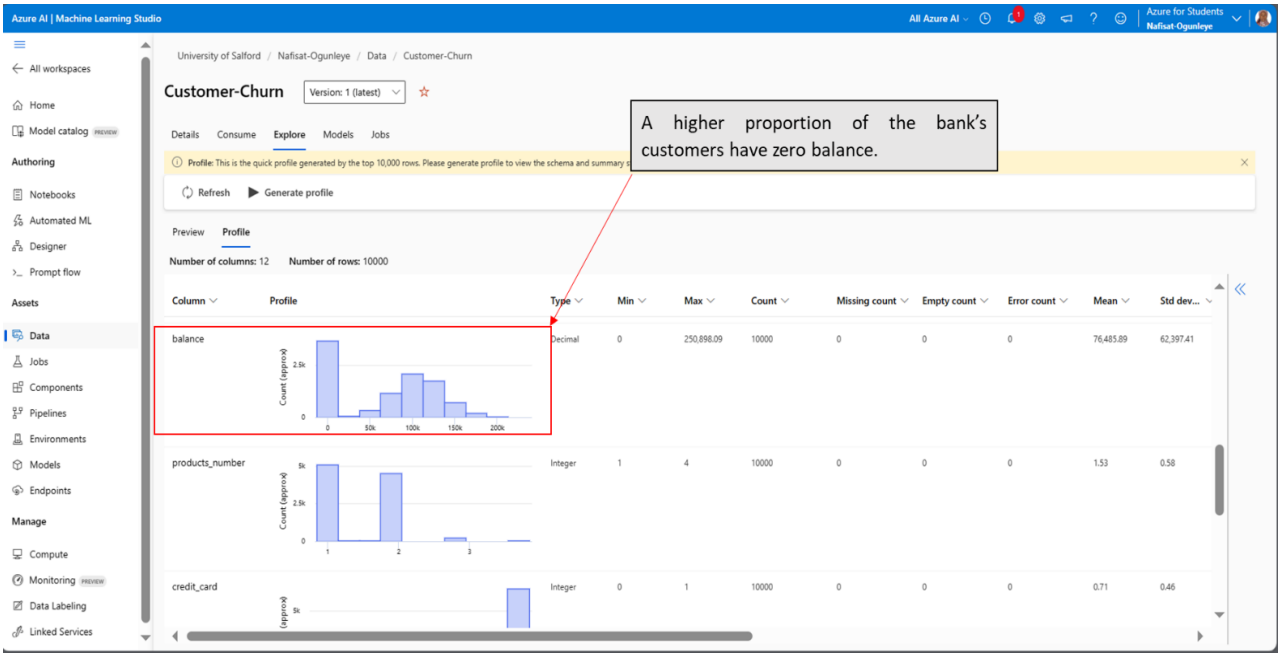


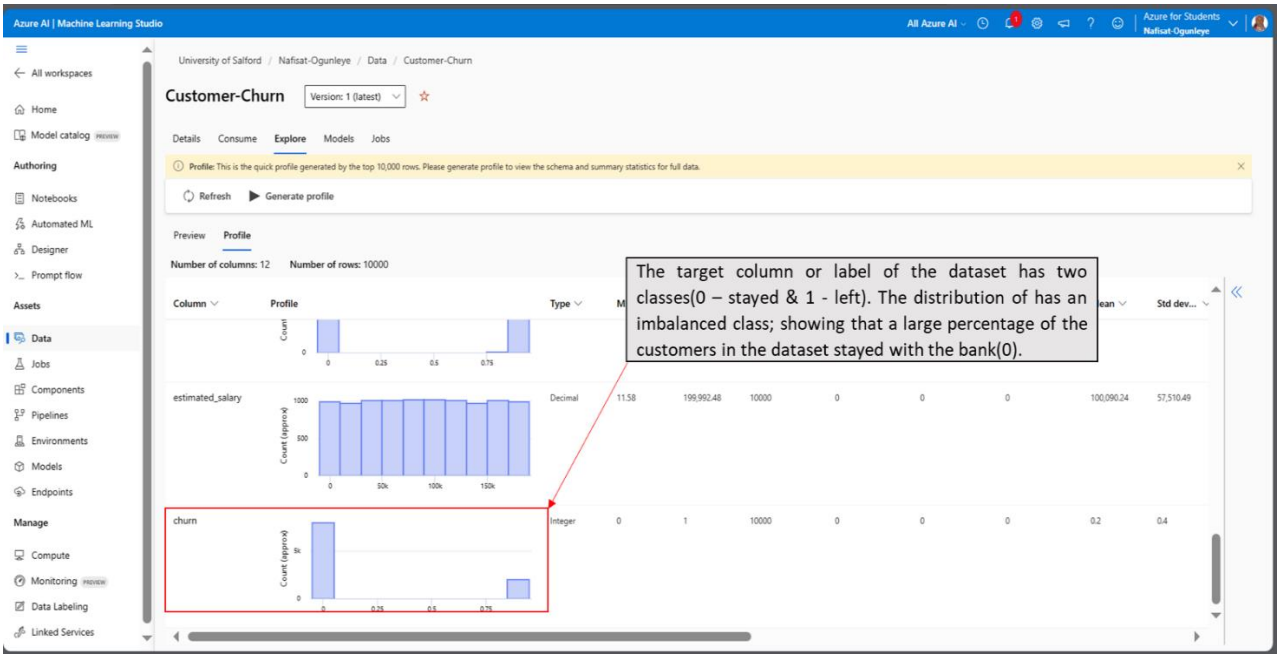
4. Following the creating of the data asset, the dataset was explored to view the distributions and statistical summary of each column and have a better understanding of the dataset.


Azure AI | Machine Learning Studio
All Azure AI
Azure for Students
Nafisat-Ogunleye
All workspaces
Home
Model catalog
Authoring
Notebooks
Automated ML
Designer
Prompt flow
Assets
Data
Jobs
Components
Pipelines
Environments
Models
Endpoints
Manage
Compute
Monitoring
Data Labeling
Linked Services
University of Salford / Nafisat-Ogunleye / Data / Customer-Churn
Customer-Churn
Version: 1 (latest)
Details
Consume
Explore
Models
Jobs
Refresh
Generate profile
Preview
Profile
Number of columns: 12
Number of rows: 10000
This shows that the customers in the dataset have a credit score that ranges between 350 to 850, with an average credit score of 653.52.
Column
Profile
Type
Min
Max
Count
Missing count
Empty count
Error count
Mean
Std dev...
customer_id
Integer
15565701
15815690
10000
0
0
0
15,690,940.57
71,936.19
credit_score
Integer
350
850
10000
0
0
0
650.53
96.65
country
String
France
Spain
10000
0
0
0
0
0
Germany
Count (approx)

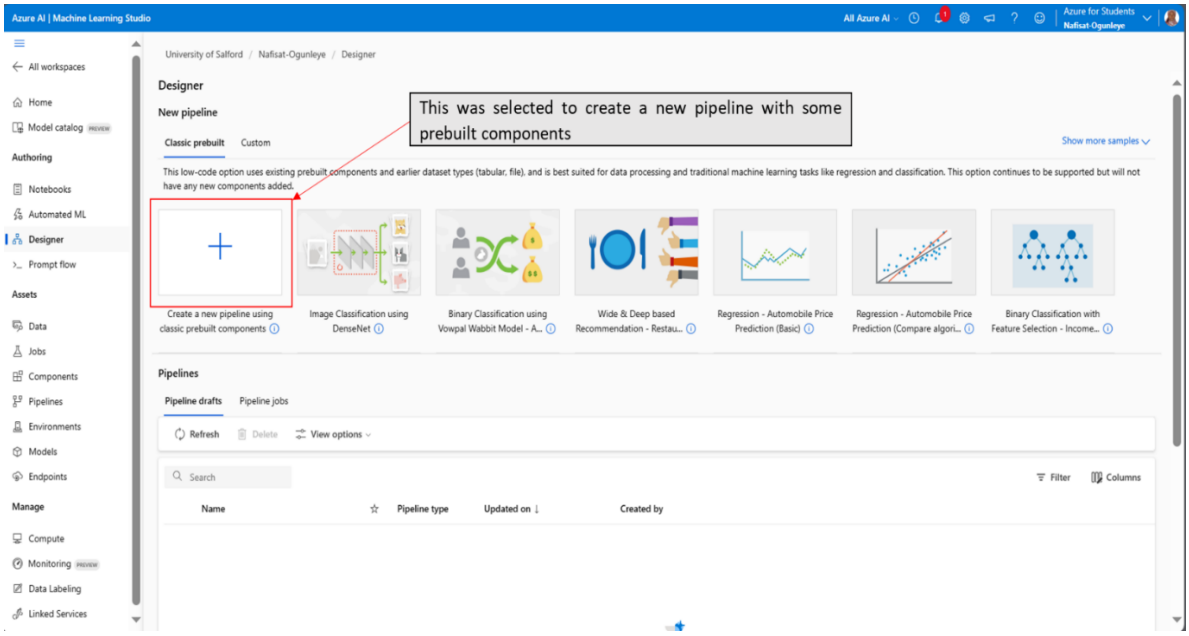

Azure AI | Machine Learning Studio
All Azure AI
Azure for Students
Nafisat-Ogunleye
All workspaces
Home
Model catalog
Authoring
Notebooks
Automated ML
Designer
Prompt flow
Assets
Data
Jobs
Components
Pipelines
Environments
Models
Endpoints
Manage
Compute
Monitoring
Data Labeling
Linked Services
University of Salford / Nafisat-Ogunleye / Data / Customer-Churn
Customer-Churn
Version: 1 (latest)
Details
Consume
Explore
Models
Jobs
Refresh
Generate profile
Preview
Profile
Number of columns: 12
Number of rows: 10000
The distribution of country shows that most of the customers are from France, which Germany and Spain having an almost equal number of customer.
This shows that the dataset has a higher number of male customers than female customers.
Column
Profile
Type
Min
Max
Count
Missing count
Empty count
Error count
Mean
Std dev...
country
String
France
Germany
Spain
Count
gender
String
Female
Male
10000
Count
age
Integer
18
92
10000
0
0
0
38.92
10.49
Unit (approx)

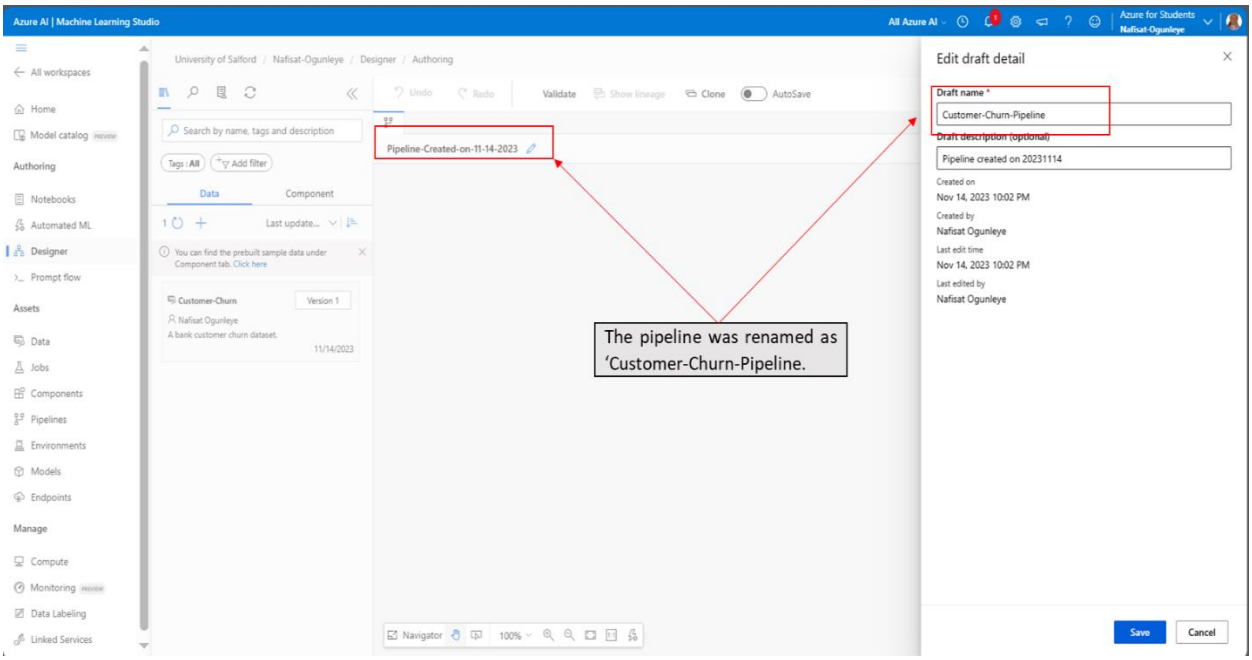

Azure AI | Machine Learning Studio
All Azure AI
Azure for Students
Nafisat-Ogunleye
All workspaces
Home
Model catalog
Authoring
Notebooks
Automated ML
Designer
Prompt flow
Assets
Data
Jobs
Components
Pipelines
Environments
Models
Endpoints
Manage
Compute
Monitoring
Data Labeling
Linked Services
University of Salford / Nafisat-Ogunleye / Data / Customer-Churn
Customer-Churn
Version: 1 (latest)
Details
Consume
Explore
Models
Jobs
Refresh
Generate profile
Preview
Profile
Number of columns: 12
Number of rows: 10000
Column
Profile
Type
Min
Max
Count
Missing count
Empty count
Error count
Mean
Std dev...
balance
Decimal
0
250,898.09
10000
0
0
0
76,485.89
62,397.41
products_number
Integer
1
4
10000
0
0
0
1.53
0.58
credit_card
Integer
0
1
10000
0
0
0
0.71
0.46
A higher proportion of the bank's customers have zero balance. The statistical description also shows that the minimum balance is 1 and the maximum is 250, 898.


Azure AI | Machine Learning Studio
All Azure AI
Azure for Students
Nafisat-Ogunleye
All workspaces
Home
Model catalog
Authoring
Notebooks
Automated ML
Designer
Prompt flow
Assets
Data
Jobs
Components
Pipelines
Environments
Models
Endpoints
Manage
Compute
Monitoring
Data Labeling
Linked Services
University of Salford / Nafisat-Ogunleye / Data / Customer-Churn
Customer-Churn
Version: 1 (latest)
Details
Consume
Explore
Models
Jobs
Refresh
Generate profile
Preview
Profile
Number of columns: 12
Number of rows: 10000
Column
Profile
Type
Min
Max
Count
Missing count
Empty count
Error count
Mean
Std dev...
balance
Decimal
0
250,898.09
10000
0
0
0
76,485.89
62,397.41
products_number
Integer
1
4
10000
0
0
0
1.53
0.58
credit_card
Integer
0
1
10000
0
0
0
0.71
0.46
A higher proportion of the bank's customers have zero balance.

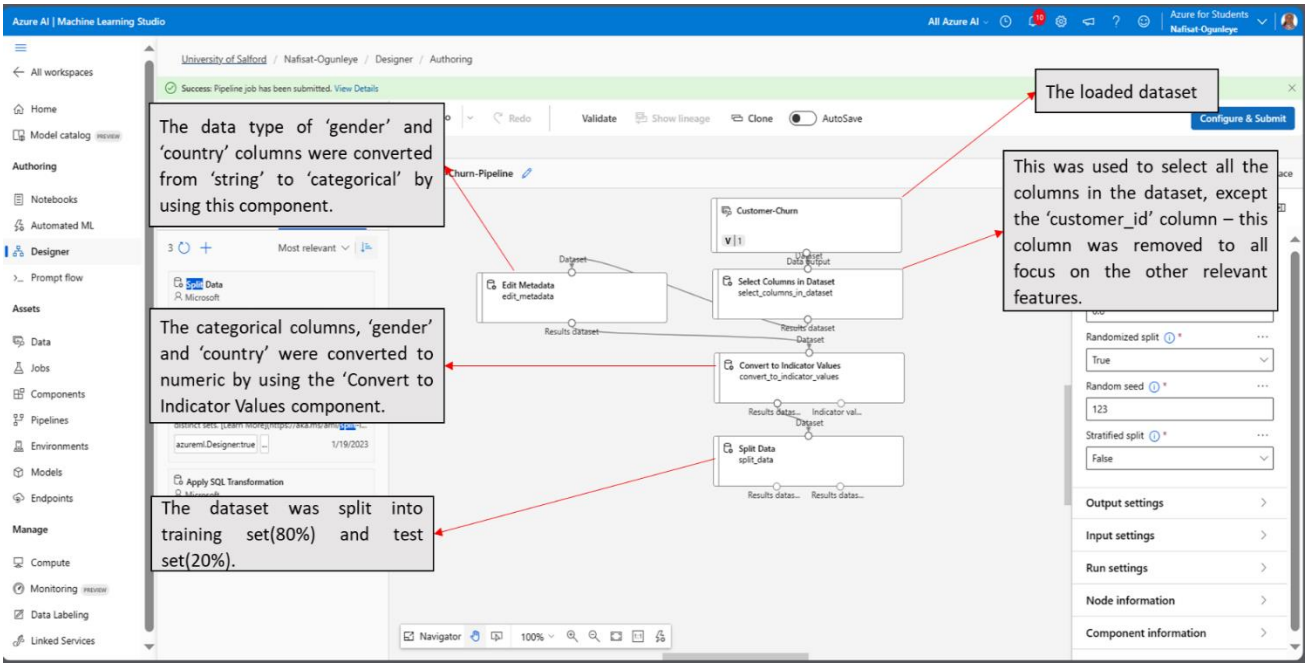


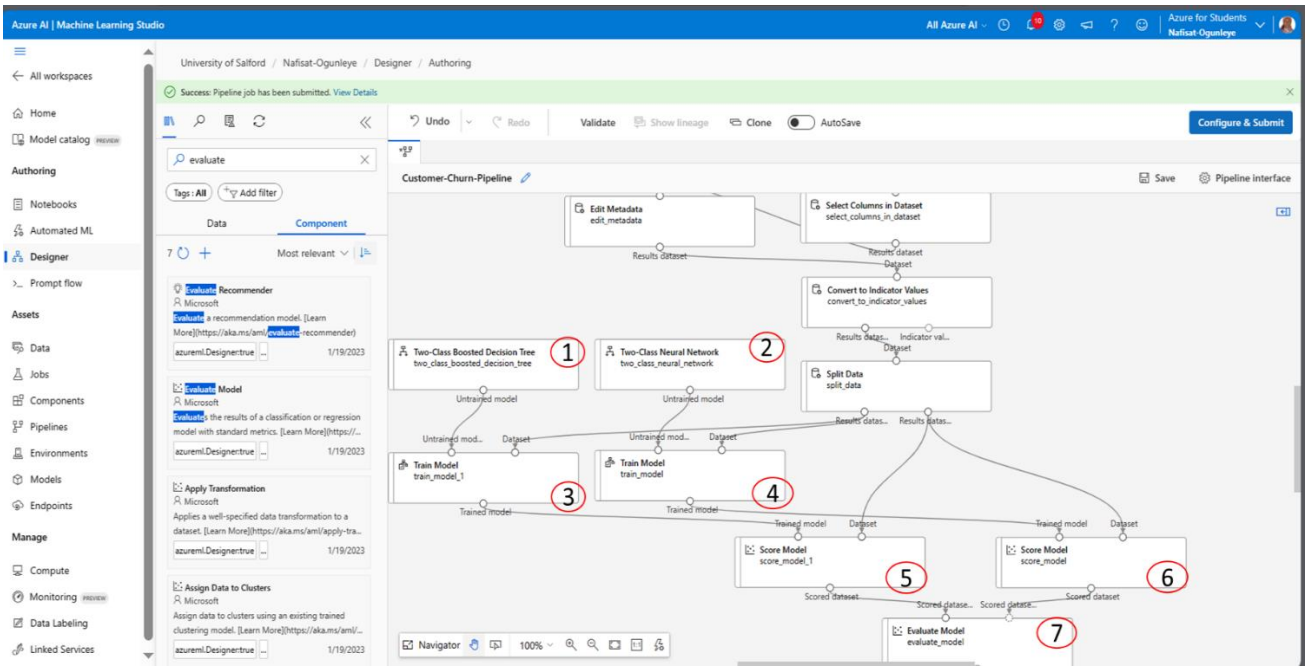
5. In the next steps, I created the Customer-Churn pipeline, loaded the dataset onto the canvas and explored two classification algorithms on the dataset.





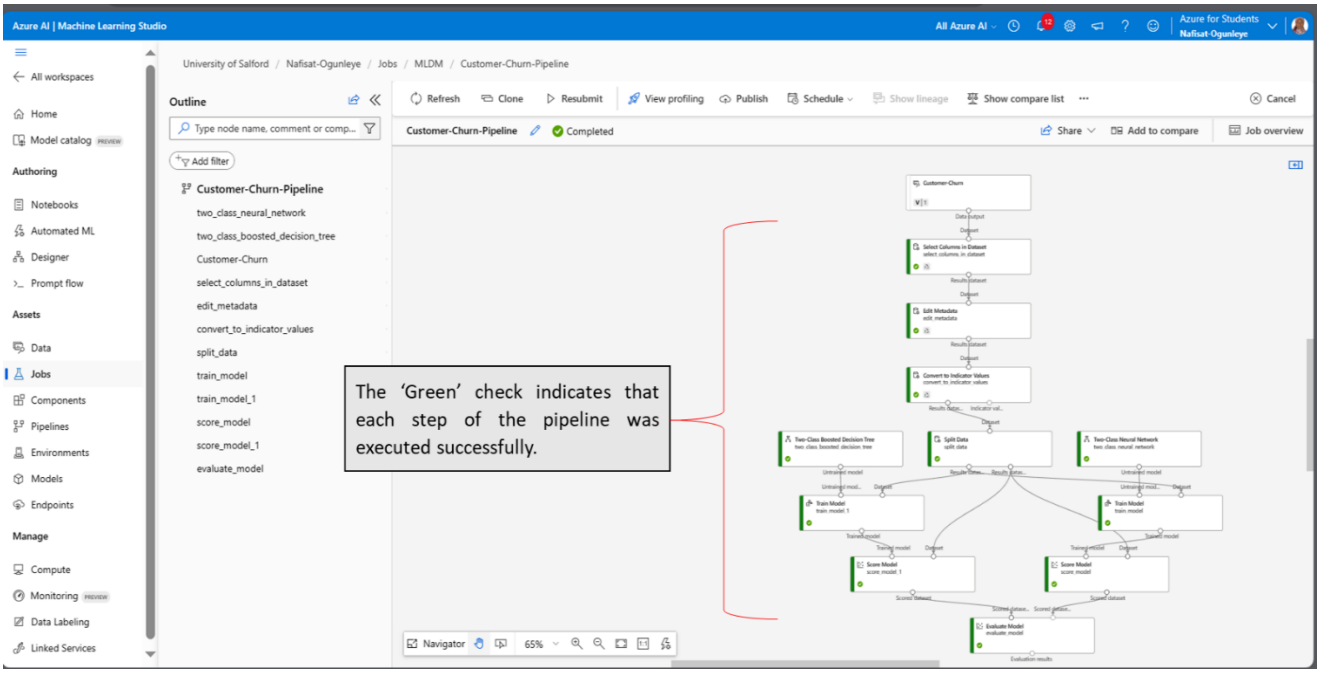
6. The image below shows the completed pipeline after transforming the dataset, applying the classification algorithms, and evaluating the performance of the algorithms.

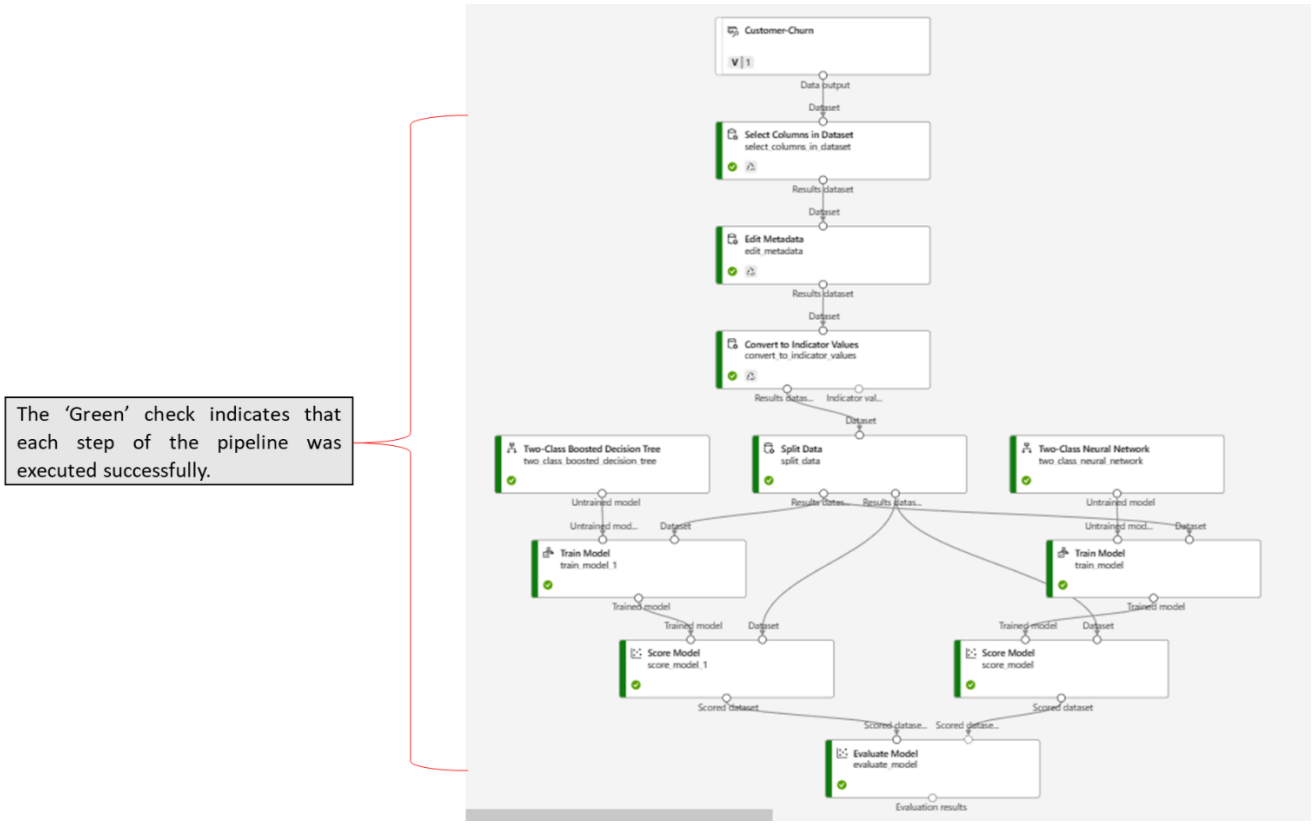




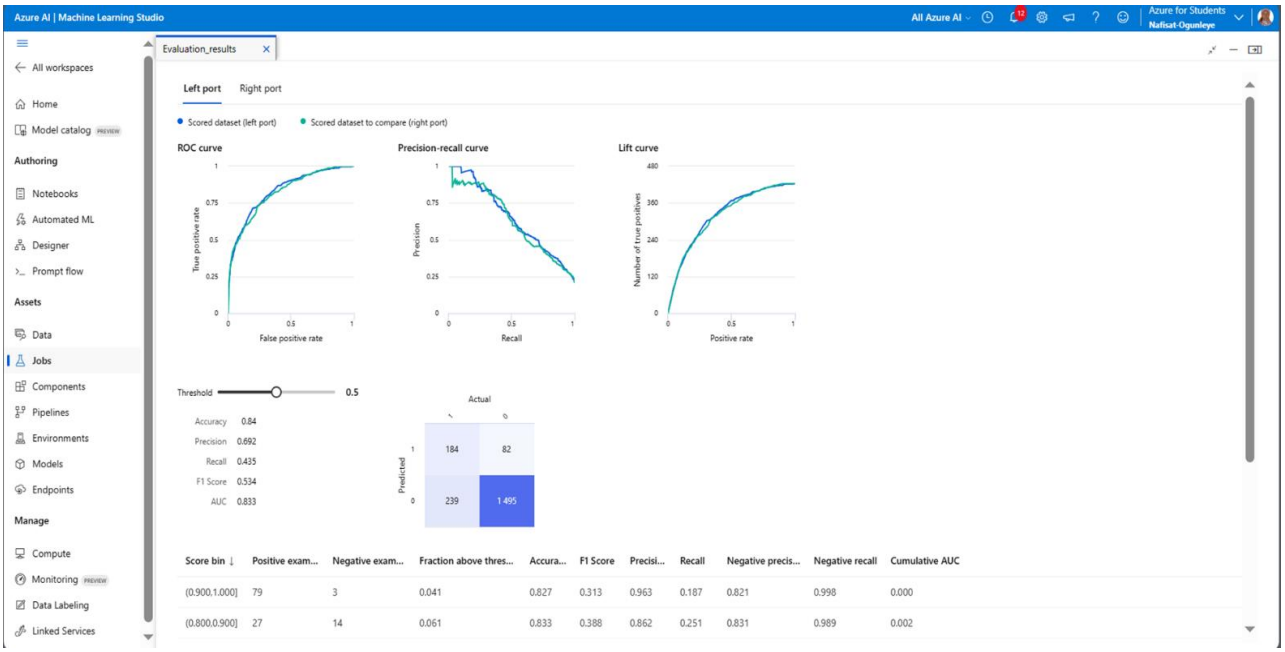
(1) & (2) represents the chosen two-class algorithms – this is because the target column had 2 classes (0 and 1). In steps (3) & (4), the two classification models were trained by using the 'Train Model' component.

(5) & (6) are the Score Model components that used the trained model dataset to make predictions on the test dataset. (7) is the final step that was used to evaluate the performance of the models.



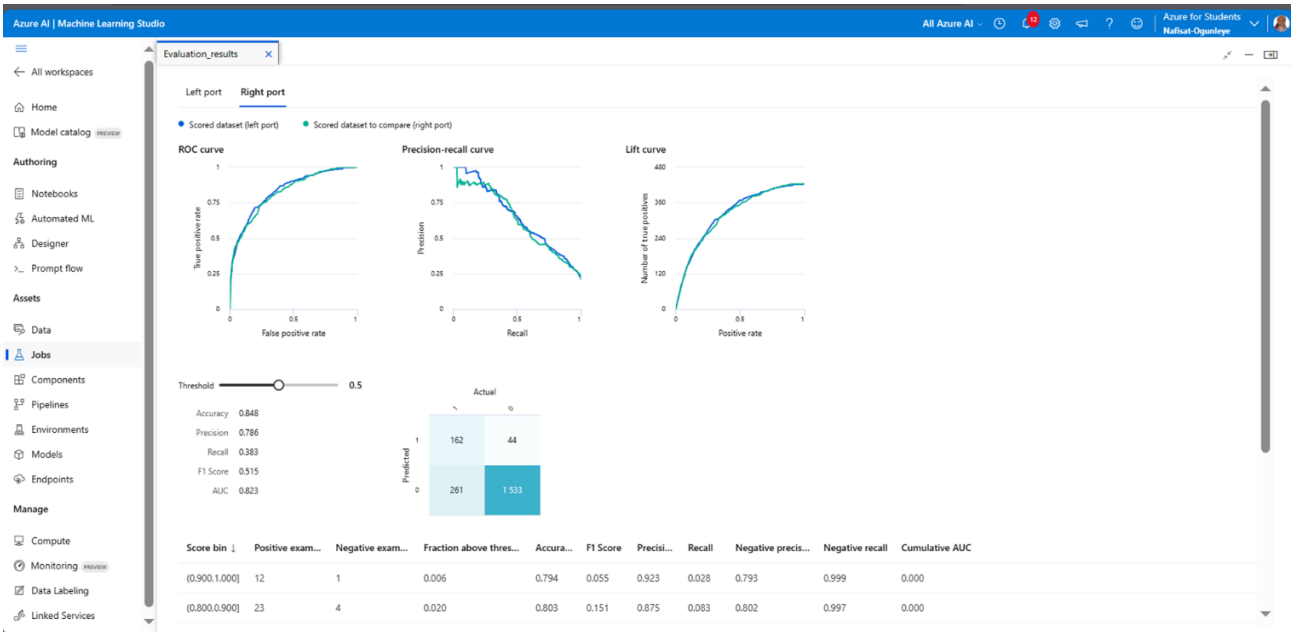


7. The result of the evaluation model was previewed to know how well the model has performed.



With a 0.5 threshold, the Two-Class Boosted Decision Tree model has an accuracy score of 0.84. This implies that the model was able to predict 84% of the test dataset correctly. A precision score of 0.692 indicates a 69% accuracy in predicting customers that stayed, while the recall score of 0.435 implies that the model captured 43.5% of customers who actually churned.

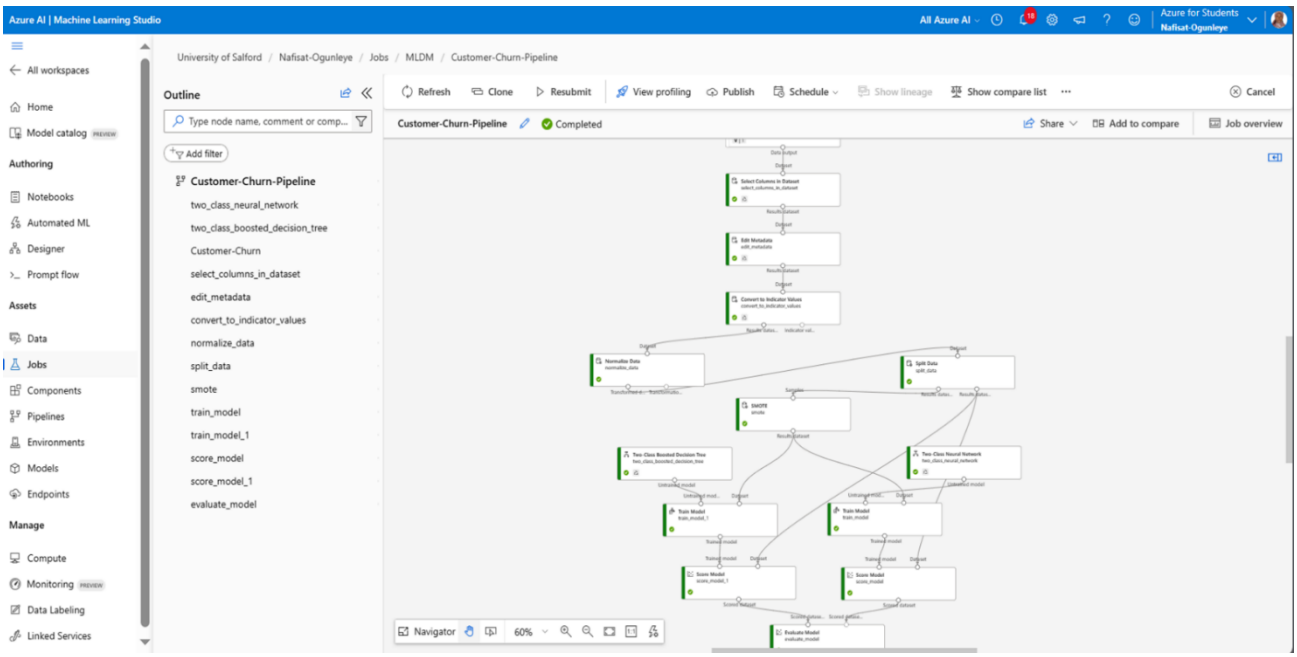
The confusion matrix output revealed that the achieved 1495 true positives(TP), 239 false positives(FP), 82 false negatives(FN) and 184 true negatives(TN).

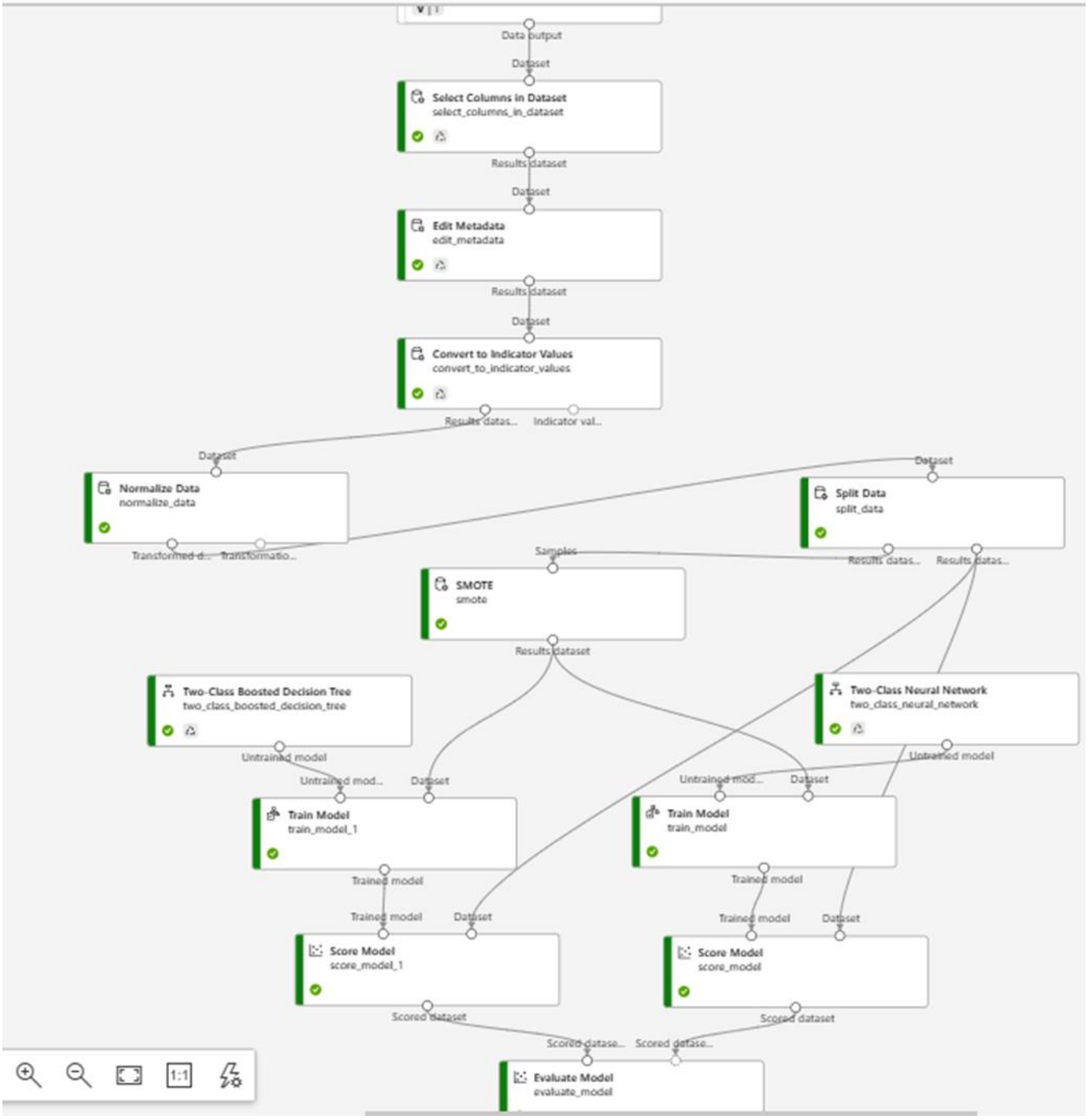


With a threshold of 0.5, the Two-Class Neural Network model predicted 84.8% of the test dataset correctly. A precision score of 0.786 indicates a 78.6% accuracy in predicting customers that stayed, while the recall score of 0.383 implies that the model captured 38.3% of customers who actually churned.

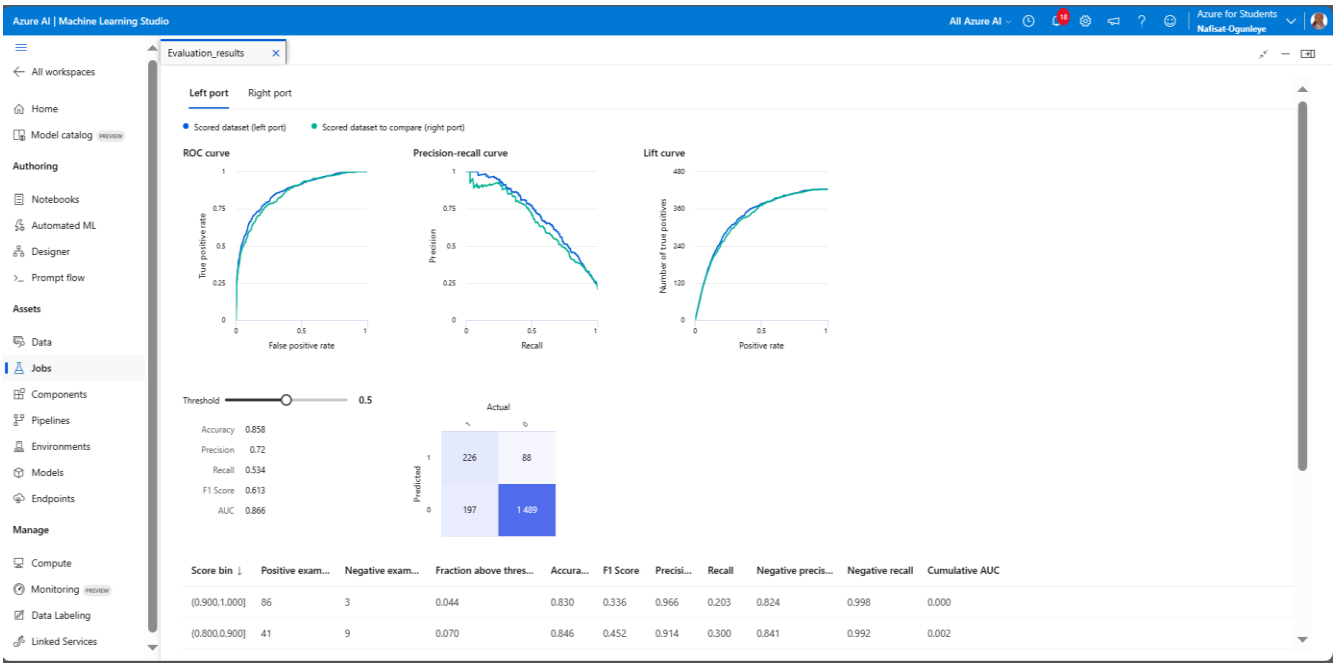
The confusion matrix output revealed that the achieved 1533 true positives(TP), 261 false positives(FP), 44 false negatives(FN) and 162 true negatives(TN).

8. To test the performance of the model regarding bias and extreme values, the dataset was normalized, using MinMax scaler, and SMOTE was used to balance the imbalanced target columns.



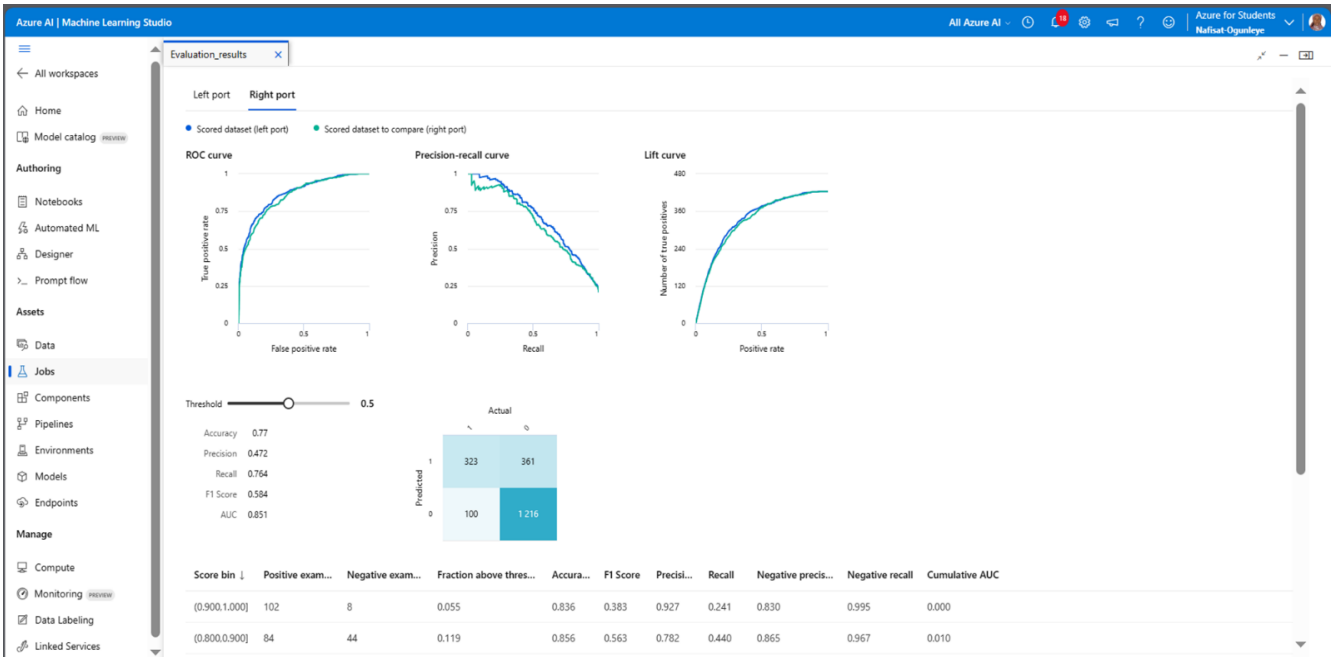


The below shows the evaluation Results of the models after using MinMax Scaler and SMOTE.



With the application of MinMax Scaler and SMOTE to the dataset, the Two-Class Boosted Decision Tree model was able to predict 85.8% of the test dataset correctly. The model has a precision score of 0.72 indicates a 72% accuracy in predicting customers that stayed, while the recall score of 0.534 implies that the model captured 53.4% of customers who actually churned.

The confusion matrix output revealed that the achieved 1489 true positives(TP), 197 false positives(FP), 88 false negatives(FN) and 226 true negatives(TN).

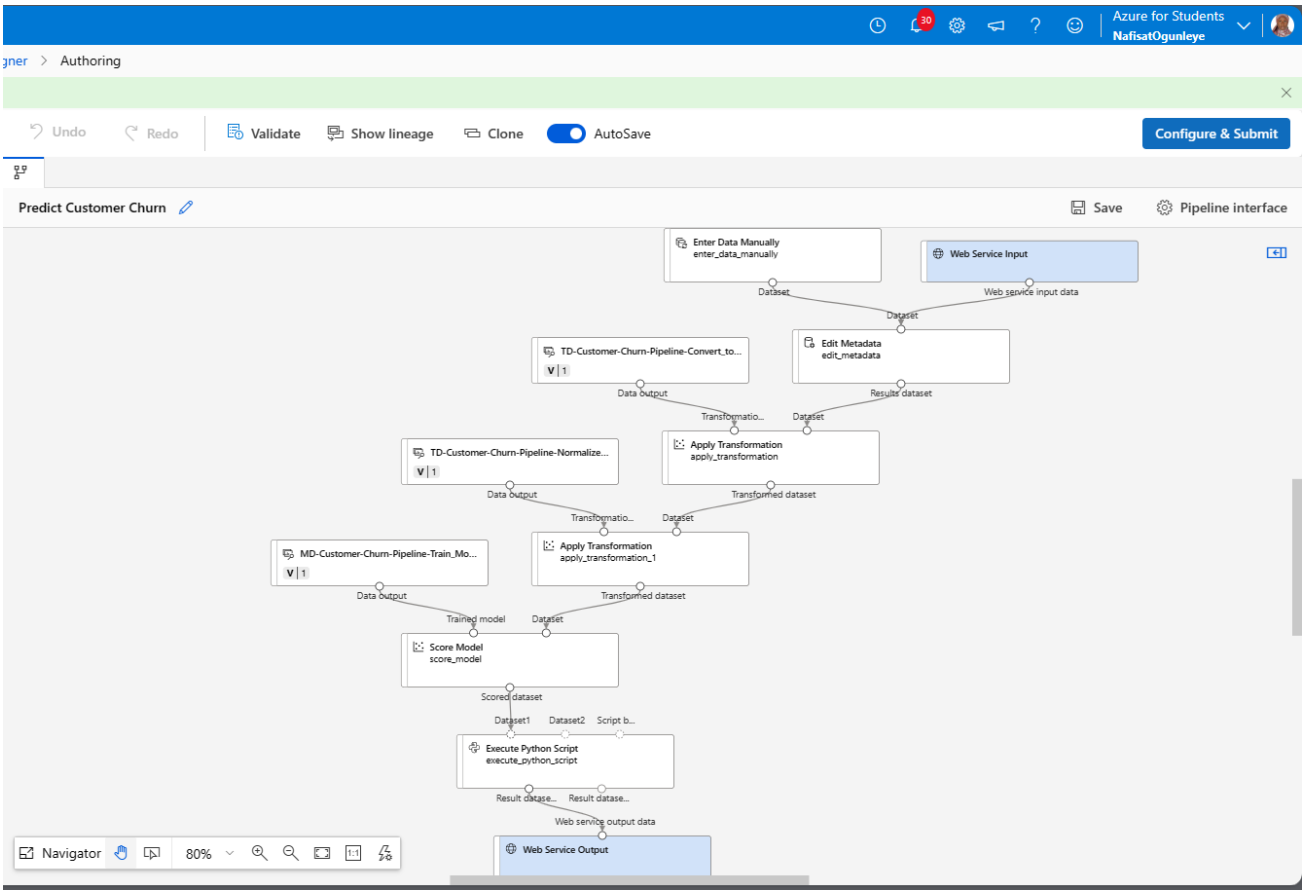


With a normalized dataset, a balanced class, and a threshold of 0.5, the Two-Class Neural Network model predicted only 77% of the test dataset correctly. It has a precision score of 0.472 indicating a 47% accuracy in predicting customers that stayed, while the recall score of 0.764 implies that the model captured 76.4% of customers who actually churned.

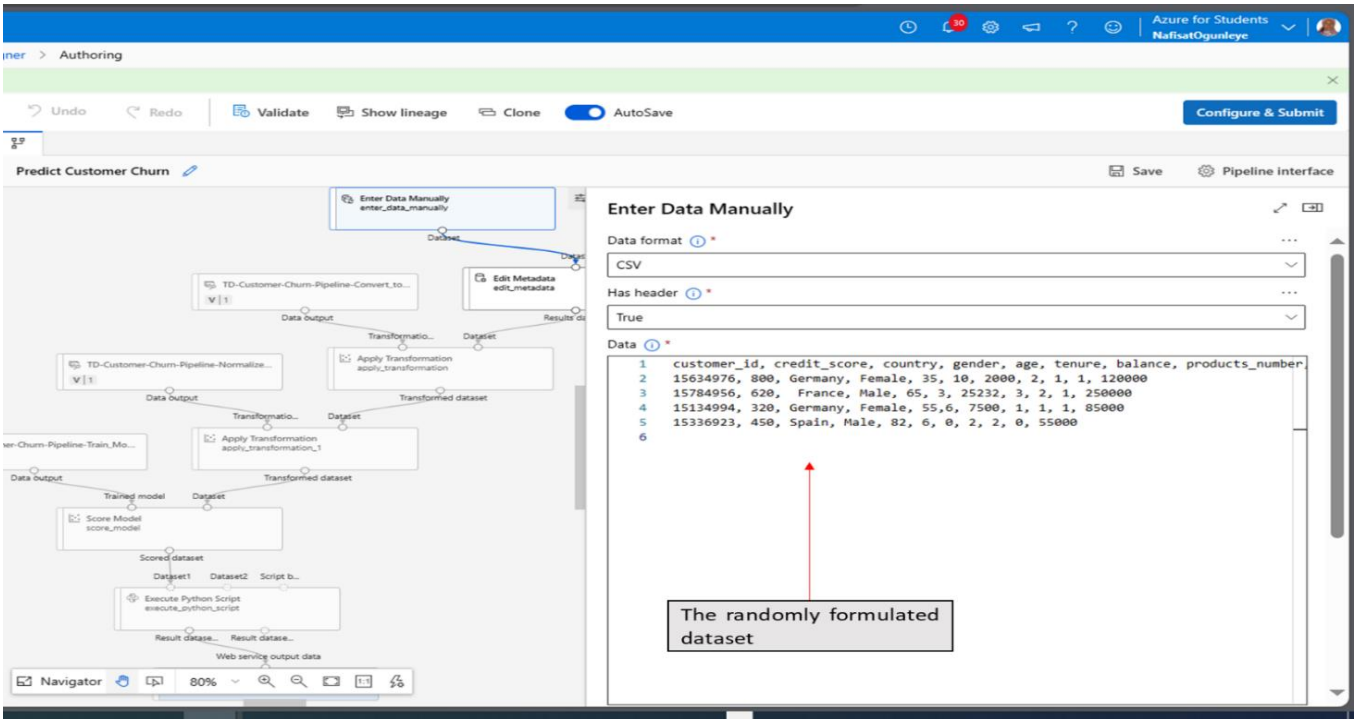
The confusion matrix output revealed that the model achieved 1216 true positives(TP), 361 false negatives(FN), 100 false positives(FP) and 323 true negatives(TN).

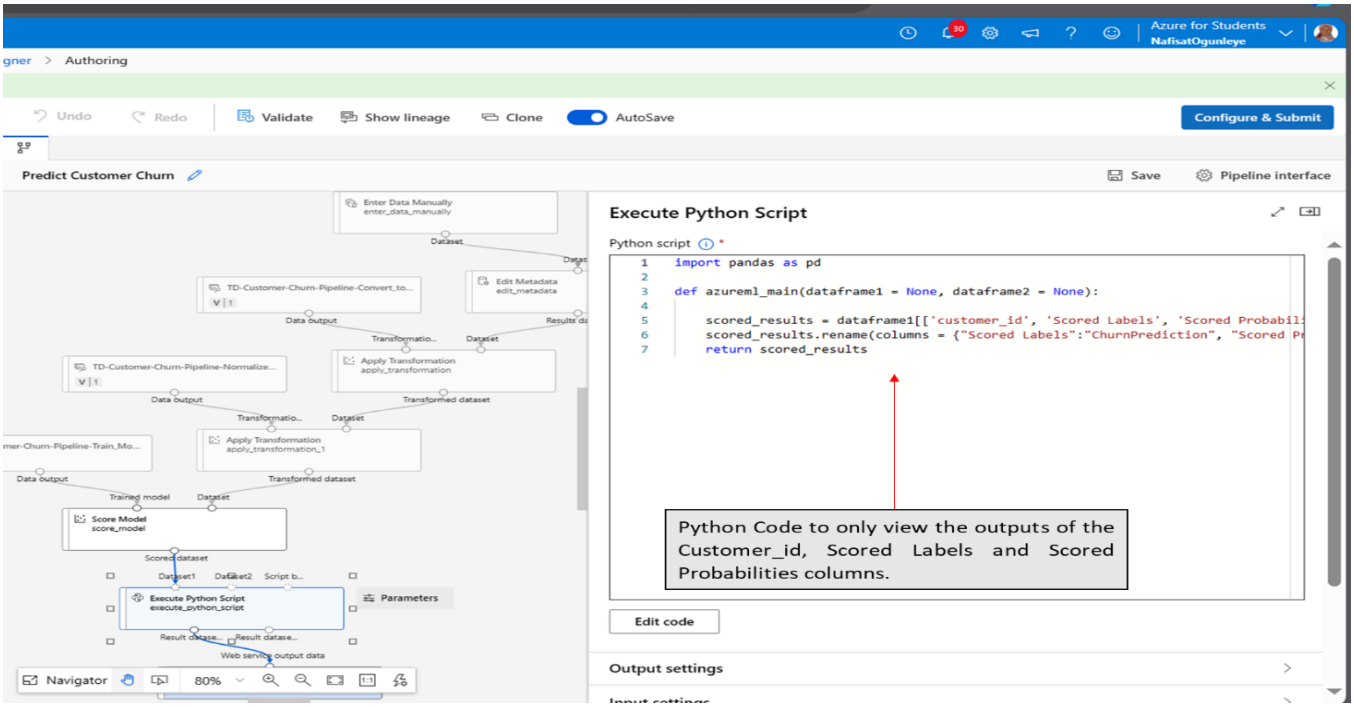
Inference Pipeline

Using the above pipeline, the two-class neural network model was removed and only the trained two-boosted decision tree model was used to create a real-time inference pipeline to predict new customers churn. The image below shows all the steps of the real-time inference pipeline.

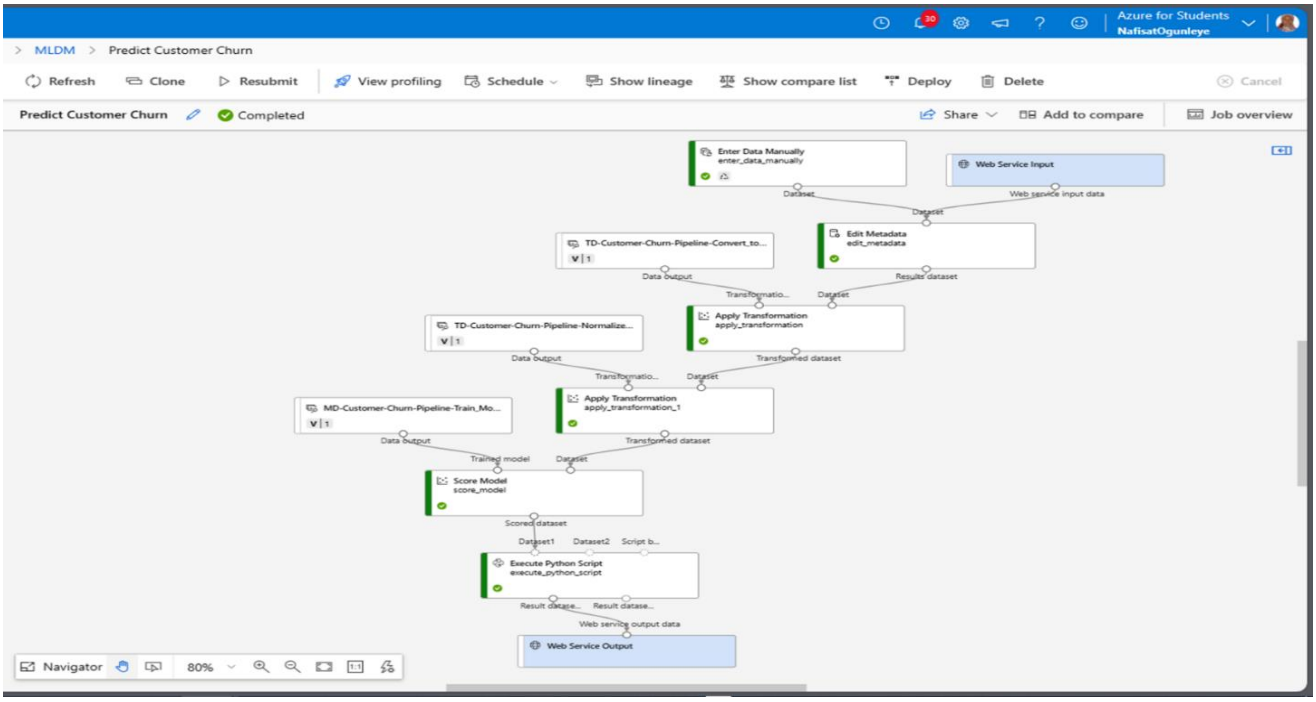


For the ‘Enter Data Manually’ step, I randomly created values for all the features that are required to predict customer churn, while ‘Execute Python Script’ was used to write the a python code to specify the columns that should be viewed from the ‘Score Model’ output.

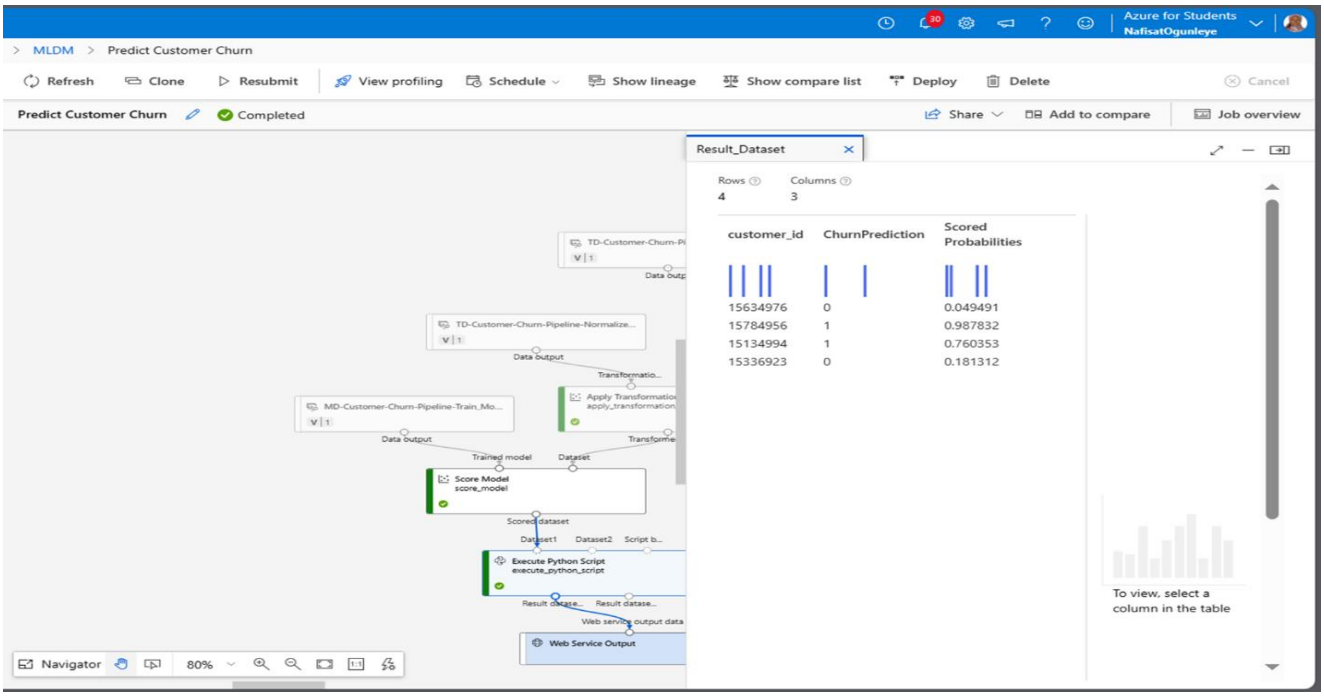




The image below shows the completed jobs of the real-time inference pipeline, and the green mark indicates that all steps were successfully executed.



The output of the executed python script was previewed to revealed the model's predictions for the manually formulated customers.



The model showed high probabilities for customers that have been predicted to churn.

Results Analysis and Discussion

Table of results for Implementation in Azure Machine Learning Designer:

Model	Evaluation Metrics	Unnormalized and Imbalanced data	Normalized and Balanced data
Two-Class Boosted Decision Tree	Accuracy	84%	85.8%
	Precision	69.2%	72%
	Recall	43.5%	53.4%
Two-Class Neural Network	Accuracy	84.8%	77%
	Precision	78.6%	47%
	Recall	38.3%	76.4%

The table above compares the performance of Two-Class Boosted Decision Tree and Two-Class Neural Network model on unnormalized and imbalanced data versus normalized and balanced data. The accuracy of the Boosted Decision Tree model improved from 84% to 85.8%, with an increased precision from 69.2% to 72% and a recall of 43.5% to 53.4%. Conversely, Neural Network's accuracy decreased from 84.8% to 77%, with a decreased precision from 78.6% to 47% and an increased recall from 38.3% to 76.4%.

While the two models have high accuracies, the Boosted decision tree model was observed to have performed better than the Neural Network model on the normalized and balanced dataset.

5. Conclusion

In conclusion, this work aimed to find the best classification algorithm to predict bank customers churn. The evaluation of two different classifiers for the bank customer churn dataset reveals that the Random Forest model outperformed the K-Nearest Neighbors (KNN) model in terms of accuracy, precision, and recall. With an higher accuracy score (84%) and a better balance between true positives and false positives, it is more effective in predicting

customer churn.

Similarly, the Two-Class Boosted Decision Tree performed better than the Two-Class Neural Network on the normalized and balanced dataset, exhibiting higher accuracy (85.8%), precision (72%), and recall (53.4%). The Boosted Decision Tree showed a better performance in capturing both churn and non-churn customers.

Overall, these findings emphasize the importance of algorithm selection and data preprocessing techniques in optimizing predictive models for predicting customers attrition and enhancing customer retention in the banking sector.