Machine Learning and Data Mining Assessment

Name	**Modupeolu Gidado**
Student ID	**@00687531**
School Name	**University Of Salford**
Date	**03/11/2023**

Task 1: Classification Algorithms for predicting Nursery Applications

Introduction:

Parents of young children are usually still in their prime and at ages where they still must work. Some, having to travel to their various places of work while others may work from home or be home makers. In which ever case, the children will have to start school at some stage. This necessitates application to nursery schools which may be accepted or rejected. This research work was carried in a location in Slovenia- Ljubljana where a particular school had so many applications and needed an objective reason to reject some of them. It was carried out in the 1980s when there was a massive enrolment to schools with a number of rejected cases. These rejections needed an objective explanation because the parents needed the help of the schools. The decision to accept or reject an application was dependent upon 3 factors: 1) The parents' occupation and the child's nursery at home, 2) The structure of the child's family and their financial standing, 3) The social and health outlook of the family. The parents' occupation said a lot about their finance and the kind of nursery the child had at home which affected the child's performance at nursery school. The family structure spoke to the family size, the number of wives and children. The finance of the home largely depended on the size of the family. A smaller sized family will averagely have better finances than a larger one. The social and health outlook depended upon the environment the family lived in, the cleanliness of the environment and the kind of people they socialised with. Based on the extracted dataset, we will evaluate the rejection of applications based on the factors itemised above. We will use two classification algorithms techniques to choose which best fits our model.

Explanation and Preparation of Dataset

For this task, the dataset was extracted from UCI Machine Learning Repository. It was created by Vladislav Rajkovic in 1990. It has 12,960 instances and 8 features. The database was developed for the demonstration of expert decision making and ranking of applications for nursery schools. The database contains eight (8) features- parents, Has_Nurs, Form, Children, Housing, Finance, Social and Health. These features are all used with an emphasis on the class attribute being what we use to analyse and make predictions on the dataset. All the features are used in the dataset with more emphasis on the categorical variable- in this case, the class feature. This is because with it, analysis and predictions are made.

Using the pandas library, the dataset was read directly from the UCI ML repository website, the headers attached and exported to a CSV file in order to be accessed easily on Jupyter Notebook.

Attribute information of the dataset

S/N	Features	Data Type
1	Parents	String
2	Has_Nurs	String
3	Form	String
4	Children	String
5	Housing	String
6	Finance	String
7	Social	String
8	Health	String
9	Class	String

To begin exploring the dataset, the following libraries were imported on to Jupyter using the codes below. It simply saying that from the filepath where the nursery data is, define the headers for the file not assuming them. Read the file into a pandas Dataframe using the specified column headers and display the first 5 rows.

```
In [2]:  import pandas as pd

         # Provide the correct file path to the car.data file
         file_path = r"C:\Users\modup\Downloads\nursery\nursery.data"

         # Define the column headers
         headers = ["parents", "has_nurs", "form", "children", "housing", "finance", "social","health", "class"]

         # Read the CSV file into a pandas DataFrame with specified column headers
         df = pd.read_csv(file_path, names=headers)

         # Display the first few rows of the DataFrame
         df.head()
```

Out[2]:

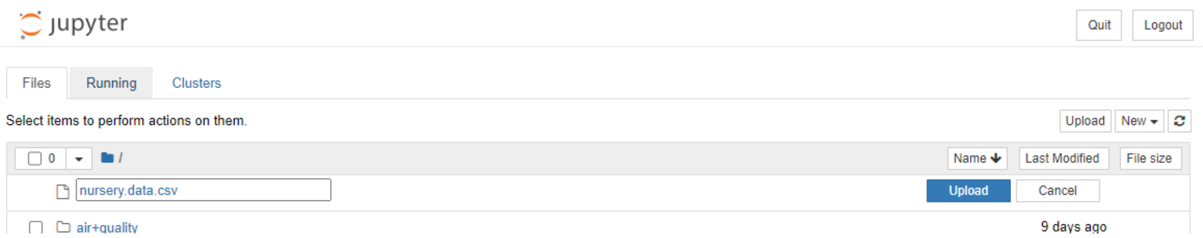
	parents	has_nurs	form	children	housing	finance	social	health	class
0	usual	proper	complete	1	convenient	convenient	nonprob	recommended	recommend
1	usual	proper	complete	1	convenient	convenient	nonprob	priority	priority
2	usual	proper	complete	1	convenient	convenient	nonprob	not_recom	not_recom
3	usual	proper	complete	1	convenient	convenient	slightly_prob	recommended	recommend

Exporting the dataframe into an Excel file from the downloaded file. This is done to easily read the file.

Using the pandas dataframe, we want to export to a csv file with the file path where it will be saved; index=False means that the Dataframe's index should not be included in the csv file so that a new column is not added to the file.

```
In [6]:  # To export the datafrme to Excel file#

         excel_output_path = r"C:\Users\modup\Downloads\nursery\nursery.data.xlsx"
         # to provide the desired output

         df.to_excel(excel_output_path, index=False)
```

After downloading the file on to the local system, it was extracted in excel and csv format before being uploaded on to the Jupyter notebook.



To read the uploaded file in csv format using pandas dataframe, display the uploaded and saved dataset.

In [28]:
```
# Uploading the saved dataset on to the local system reading it with panda

df = pd.read_csv("nursery.data.csv")
df.head()
```

Out[28]:

	parents	has_nurs	form	children	housing	finance	social	health	class
0	usual	proper	complete	1	convenient	convenient	nonprob	recommended	recommend
1	usual	proper	complete	1	convenient	convenient	nonprob	priority	priority
2	usual	proper	complete	1	convenient	convenient	nonprob	not_recom	not_recom
3	usual	proper	complete	1	convenient	convenient	slightly_prob	recommended	recommend
4	usual	proper	complete	1	convenient	convenient	slightly_prob	priority	priority

In [29]:
```
df.tail()
```

Out[29]:

	parents	has_nurs	form	children	housing	finance	social	health	class
12955	great_pret	very_crit	foster	more	critical	inconv	slightly_prob	priority	spec_prior

In order to get a brief description of the dataframe, they df.describe syntax is used. Here, it can be seen that the whole dataset comprises of 12,960 instances, the unique value of each feature, the top parameters and the most frequent of them.

[30]:
```
df.describe()
```

Out[30]:

	parents	has_nurs	form	children	housing	finance	social	health	class
count	12960	12960	12960	12960	12960	12960	12960	12960	12960
unique	3	5	4	4	3	2	3	3	5
top	usual	proper	complete	1	convenient	convenient	nonprob	recommended	not_recom
freq	4320	2592	3240	3240	4320	6480	4320	4320	4320

To confirm that there are or there no missing values, df.isnull().sum()

```
In [11]:  # To confirm if there are missing values
          df.isnull().sum()

Out[11]:  parents     0
          has_nurs    0
          form        0
          children    0
          housing     0
          finance     0
          social      0
          health      0
          class       0
          dtype: int64
```

The df.shape syntax describes the number of rows and columns in the dataset. Here, there are 9 columns earth 12,960 rows.

```
In [31]:  df.shape

Out[31]:  (12960, 9)
```

For df.info syntax, it is telling us more about the features of the dataset. This includes the number of columns, the label of each column, the count of null values, the datatypes and the memory usage in the dataset. All the variables are objects/ string type and there are no missing values. From this point, we can download the other libraries needed to perform the analysis of the dataset.

```
In [32]:  df.info()

          <class 'pandas.core.frame.DataFrame'>
          RangeIndex: 12960 entries, 0 to 12959
          Data columns (total 9 columns):
           #   Column    Non-Null Count  Dtype
          ---  ------    --------------  -----
           0   parents   12960 non-null  object
           1   has_nurs  12960 non-null  object
           2   form      12960 non-null  object
           3   children  12960 non-null  object
           4   housing   12960 non-null  object
           5   finance   12960 non-null  object
           6   social    12960 non-null  object
           7   health    12960 non-null  object
           8   class     12960 non-null  object
          dtypes: object(9)
          memory usage: 911.4+ KB
```

Implementation in Python/ Jupyter Notebook

To implement any algorithm technique on the dataset, the following libraries were imported.

In [33]:

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

And because, the datatype is on string format, we must convert the categorical values to numeric values so that the analysis can be read and understood using the code below. Each object of an attribute has been assigned a numeric value. For instance, 'usual parents' are assigned '2', if a child has a proper nursery, he is assigned '3' and so on. The df.head syntax confirms the assignment.

In [36]:

```
#Converting categorical values to numeric values

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df["parents"]=le.fit_transform(df["parents"])
df["has_nurs"]=le.fit_transform(df["has_nurs"])
df["form"]=le.fit_transform(df["form"])
df["children"]=le.fit_transform(df["children"])
df["housing"]=le.fit_transform(df["housing"])
df["finance"]=le.fit_transform(df["finance"])
df["social"]=le.fit_transform(df["social"])
df["health"]=le.fit_transform(df["health"])
df["class"]=le.fit_transform(df["class"])

df.head()

# confirm that the values have been converted to numeric
```

Out[36]:

	parents	has_nurs	form	children	housing	finance	social	health	class
0	2	3	0	0	0	0	0	2	2
1	2	3	0	0	0	0	0	1	1

We use the syntax df.describe(include=all) to calculate the mean, standard deviation, min, quartiles, max and view them at a glance.

In [24]:

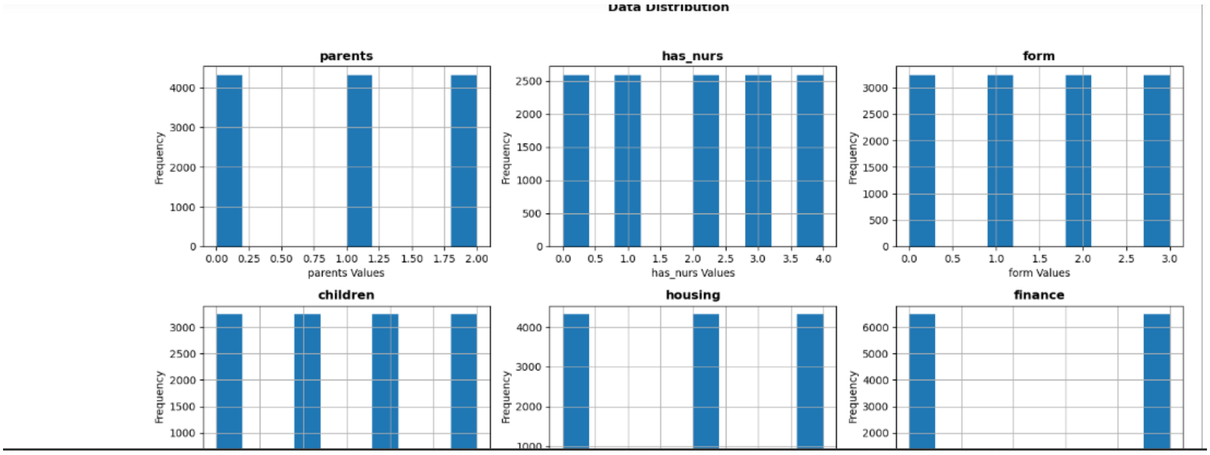
```
df.describe(include="all")
```

Out[24]:

	parents	has_nurs	form	children	housing	finance	social	health	class
count	12960.000000	12960.000000	12960.000000	12960.000000	12960.000000	12960.000000	12960.000000	12960.000000	12960.000000
mean	1.000000	2.000000	1.500000	1.500000	1.000000	0.500000	1.000000	1.000000	1.366821
std	0.816528	1.414268	1.118077	1.118077	0.816528	0.500019	0.816528	0.816528	1.294212
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	1.000000	0.750000	0.750000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	1.000000	2.000000	1.500000	1.500000	1.000000	0.500000	1.000000	1.000000	1.000000
75%	2.000000	3.000000	2.250000	2.250000	2.000000	1.000000	2.000000	2.000000	3.000000
max	2.000000	4.000000	3.000000	3.000000	2.000000	1.000000	2.000000	2.000000	4.000000

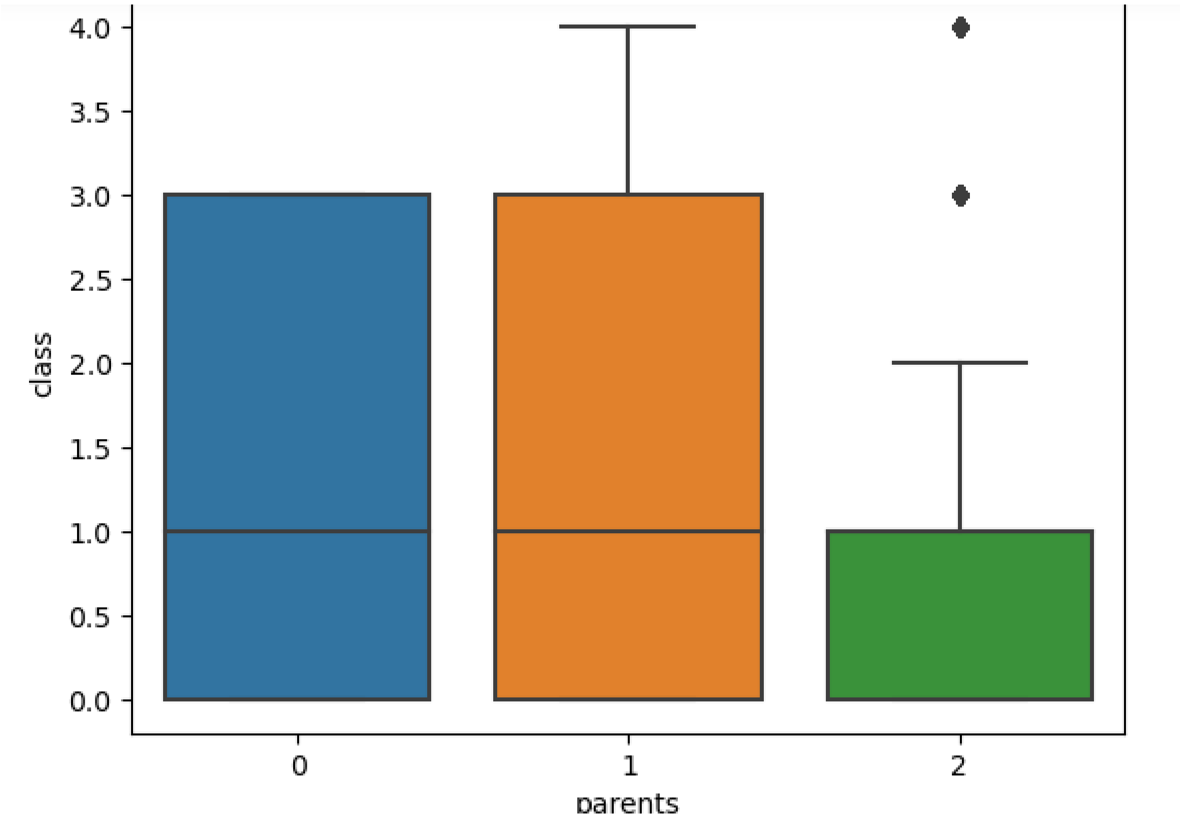
The code below is used to give a graphical description of each feature. The distribution of which shows an equal amount of each feature.

```
In [38]:  #Data description
          # Histograms for each feature to visualize data distribution
          axes = df.hist(figsize=(14, 10))

          # Adding titles and labels to each histogram
          for ax in axes.ravel():
              ax.set_title(ax.get_title(), fontweight='bold')
              ax.set_xlabel(ax.get_title() + ' Values')
              ax.set_ylabel('Frequency')

          plt.suptitle('Data Distribution', y=1.02, fontweight='bold')
          plt.tight_layout()
          plt.show()
```



Using a boxplot to display the distribution of the dataset, the blue represents gret-pret, orange represents pretentious and the green box represents usual type of parents respectively. It represents the interquartile range which is the range between the 1st and 3rd quartile. The line in the box represents the 2nd quartile which is the median (50%) of the dataset.

```
In [40]: # Assuming 'df' is your DataFrame and 'class' is the column you want to plot

         sns.boxplot(y='class', x='parents', data=df)
         plt.title('Boxplot of Feature by Class')
         plt.show()
```

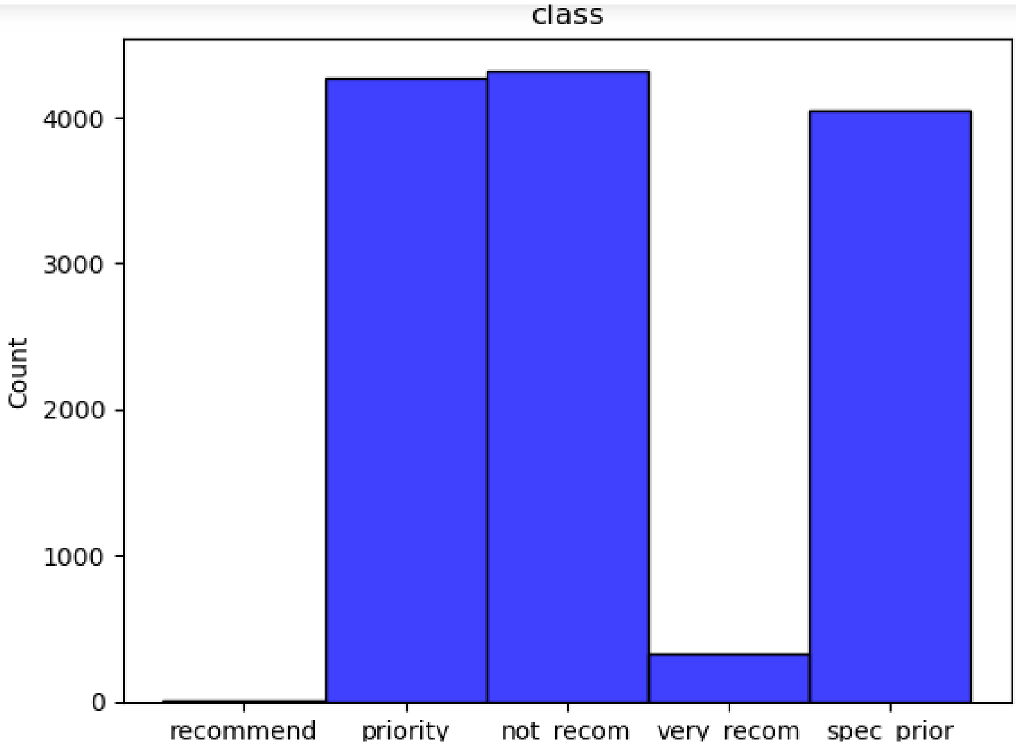


To view the numerical distribution of the target variable- class. The variables in the target are not_recom, priority, spec_prior, very_recom and recommended.

```
In [16]: df2=df["class"].value_counts()
         print(df2)

         not_recom     4320
         priority      4266
         spec_prior    4044
         very_recom     328
         recommend        2
         Name: class, dtype: int64
```

Using a graphical representation of the class distribution, we have the below

```
In [15]: # To view the distribution of the target variable (class)

         sns.histplot(df["class"], color="blue")
         plt.title("class")
         plt.show()
```



Based on the output above, it is seen the target variable is imbalanced and needs to be balanced before training the model/ algorithm. It is important to balance a categorical feature of this nature to improve the performance of the machine learning algorithm. The justification for this is to avoid any bias that will make the model assign more importance to the classes with higher instances while assigning less importance to those underrepresented. We install scikit.learn imbalanced learn to perform the balancing task.

```
In [19]: pip install scikit-learn imbalanced-learn

         Requirement already satisfied: scikit-learn in c:\users\modup\onedrive\documents\anaconda for jupyter\lib\site-packages (1.
         3.0)
         Requirement already satisfied: imbalanced-learn in c:\users\modup\onedrive\documents\anaconda for jupyter\lib\site-packages
         (0.11.0)
         Requirement already satisfied: numpy>=1.17.3 in c:\users\modup\onedrive\documents\anaconda for jupyter\lib\site-packages (fr
         om scikit-learn) (1.24.3)
         Requirement already satisfied: scipy>=1.5.0 in c:\users\modup\onedrive\documents\anaconda for jupyter\lib\site-packages (fro
         m scikit-learn) (1.10.1)
         Requirement already satisfied: joblib>=1.1.1 in c:\users\modup\onedrive\documents\anaconda for jupyter\lib\site-packages (fr
         om scikit-learn) (1.2.0)
         Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\modup\onedrive\documents\anaconda for jupyter\lib\site-packa
         ges (from scikit-learn) (2.2.0)
         Note: you may need to restart the kernel to use updated packages.
```

Applying the SMOTE technique to balance the data, we begin splitting the dataset into training and test sets, defining the dependent and independent variables- y and X respectively.

```
In [ ]:  # To split the dataset into training nand test sets and also handle class imbalance with SMOTE

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

         # Define the dependent and independent variables

         X = df.drop("class", axis = 1)
         y = df["class"]

         # Split the dataset

         X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2, stratify=y, random_state=0)

         # Apply SMOTE to the training dataset to handle class imbalance

         smote = SMOTE(sampling_strategy="auto",k_neighbors=min(1, len(y_train) - 1),random_state=0)
         X_train, y_train=smote.fit_resample(X_train, y_train)
```

After using the SMOTE technique to balance the class target variable, we view the result as follows:

```
In [21]:  y_train.value_counts()

Out[21]:  3    3456
          0    3456
          1    3456
          4    3456
          2    3456
          Name: class, dtype: int64
```

Now that the dataset is balanced, we can train our model with any algorithm. For this assessment, the algorithms used to train the model will be K- Nearest Neighbors, and Decision Tree.

a. Using KNN, the class features and the input features will be determined. From the dataset, the target variable which is the same as the dependent variable is the class label with other variables being the independent features. The purpose of this classification technique is to predict the class label of a sample of y-dependent variable based on the features of X.

To extract the independent variables X and the dependent variable y from the dataframe to further analyse the dataset.

```
n [21]:  # Defining the dependent and independent variables

         x = df.iloc[:, [0,1,2,3,4,5,6,7]].values
         y = df.iloc[:, 8].values
```

To split the dataset into training and test sets

```
In [38]: # Split the dataset into training and test

         from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2, stratify=y, random_state=0)
```

To standardise the training and test sets and fit the K-NN algorithm into the training set

```
In [23]: # For standardisation

         from sklearn.preprocessing import StandardScaler
         sc=StandardScaler()
         X_train_s=sc.fit_transform(X_train)
         X_test_s=sc.transform(X_test)
```

To fit the K-NN algorithm into the training dataset, we imported the KNeighborsClassifier from sklearn.neighbors.

```
In [125]: # Fitting KNN to the training set

          from sklearn.neighbors import KNeighborsClassifier
          classifier=KNeighborsClassifier(n_neighbors=8, metric='minkowski', p=2)
          classifier.fit(X_train, y_train)

Out[125]: KNeighborsClassifier(n_neighbors=8)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

To evaluate the model and make predictions on the test results, we can view the first 3 and last 3 predictions.

```
In [126]: #Evaluating the model and predicting the test result

          y_pred=classifier.predict(X_test)
          print(y_pred)

          [0 3 0 ... 0 1 1]
```

The result for both training and test models are the same going by the first and last 3.

```
In [127]: print(y_test)
```

```
6851     0
3013     3
3392     0
6785     0
1809     4
        ..
4113     3
10948    3
491      0
438      1
337      1
Name: class, Length: 2592, dtype: int32
```

Creating the confusion Matrix and determining the accuracy of the predictions

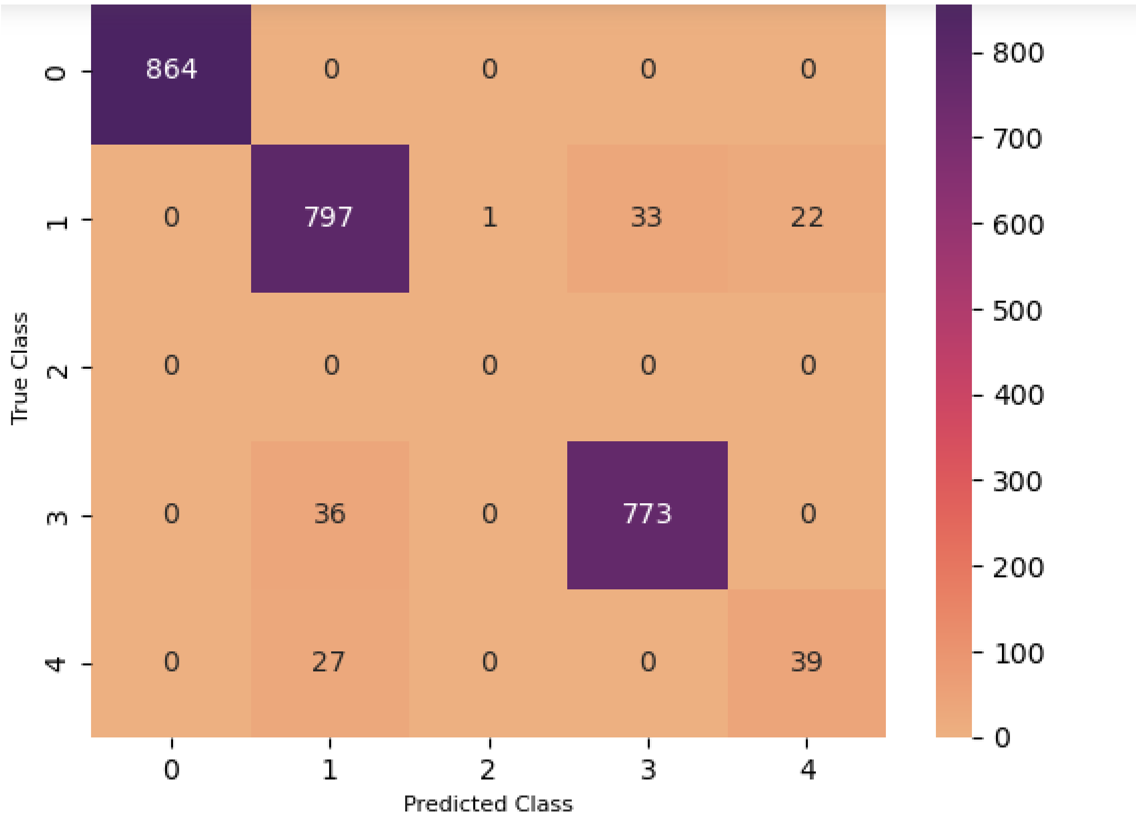
```
In [128]: from sklearn import metrics
          acc=metrics.accuracy_score(y_test, y_pred)
          print ('accuracy:%.2f\n\n'%(acc))
          cm=metrics.confusion_matrix(y_test,y_pred)
          print('Confusion Matrix:')
          print(cm,'\n\n')
          print('-------------------------------------------------------------------')
          result=metrics.classification_report(y_test,y_pred)
          print('Classification Report:\n')
          print(result)
```

```
accuracy:0.95

Confusion Matrix:
[[864   0   0   0   0]
 [  0 797   1  33  22]
 [  0   0   0   0   0]
 [  0  36   0 773   0]
 [  0  27   0   0  39]]


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

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       864
           1       0.93      0.93      0.93       853
           2       0.00      0.00      0.00         0
           3       0.96      0.96      0.96       809
           4       0.64      0.59      0.61        66

    accuracy                           0.95      2592
   macro avg       0.71      0.70      0.70      2592
weighted avg       0.95      0.95      0.95      2592
```

There is a 95% accuracy of this prediction which makes this algorithm a good one to test predictions of nursery school application based on the target variable.

Using Seaborn heatmap to visualise the confusion matrix,

```
In [27]:  ax = sns.heatmap(cm, cmap='flare',annot=True, fmt='d')
          plt.xlabel("Predicted Class", fontsize=8)
          plt.ylabel("True Class", fontsize=8)
          plt.title("Confusion Matrix", fontsize=8)

          plt.show()
```



This is a pictorial representation of the accuracy of prediction.

b. Using Decision Tree algorithm to train a model in Python, we first import the DecisionTreeClassisfier library from sklearn.tree and then the parameter was set to entropy so that the algorithm will use information gain as the criterion to make decisions. The random_state parameter was set to 0 to ensure that we get the same result each time the code is run. The last line fits the desion tree algorithm on the training data- ‘x_train’ being the independent variables and ‘y_train’ being the dependent or target variable.

```
In [28]: # Using the decision tree algorithm

         # Fitting Decision tree algorithm to the training set

         from sklearn.tree import DecisionTreeClassifier
         classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
         classifier.fit(X_train, y_train)
```

```
Out[28]:                 DecisionTreeClassifier
         DecisionTreeClassifier(criterion='entropy', random_state=0)
```

Evaluating the model and predicting the test result

```
In [29]: #Evaluating the model and predicting the test result

         y_pred=classifier.predict(X_test)
         print(y_pred)

         [0 3 0 ... 0 1 1]
```

The result of the prediction are the same as we had in the previous algorithm testing. However, testing the accuracy of the model using decision tree produces a higher accuracy of prediction- 99%.

```
In [30]: # Evaluating the perfomance of the model with some matrix
         # accuracy of the model is 99% in predicting the test set which is very good

         from sklearn import metrics
         acc=metrics.accuracy_score(y_test, y_pred)
         print ('accuracy:%.2f\n\n'%(acc))
         cm=metrics.confusion_matrix(y_test,y_pred)
         print('Confusion Matrix:')
         print(cm,'\n\n')
         print('-----------------------------------------------------------------------')
         result=metrics.classification_report(y_test,y_pred)
         print('Classification Report:\n')
         print(result)

         accuracy:0.99


Confusion Matrix:
[[864   0   0   0]
 [  0 847   3   3]
 [  0   6 803   0]
 [  0   7   0  59]]


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

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       864
           1       0.98      0.99      0.99       853
           3       1.00      0.99      0.99       809
           4       0.95      0.89      0.92        66

    accuracy                           0.99      2592
   macro avg       0.98      0.97      0.98      2592
weighted avg       0.99      0.99      0.99      2592
```

Result Analysis and Discussion of Result:

The performance metrics used to evaluate the models are the KNN and Decision tree algorithms as requested for the assessment. These two were chosen to make prediction the success or rejection of nursery school application in Slovenia. The dataset had equal distribution across variables thus taking out any form of bias.

Evaluating the results using the accuracy of both algorithms, it is evident that the decision tree gives us a better prediction of the success of a nursery school application in the area of Slovenia in thr 1980s. Giving us a 99% accuracy whereas KNN provided us with a 95% accuracy.

The main ethical, legal or professional cnsiderations in using machine learning and data mining on dataset is privacy. According to Harvard Business School online - https://online.hbs.edu/blog/post/data-ethics there are 5 principles of data ethics. These include Ownership, Transparency, Privacy, Intention and Outcomes.

Ownership: This is the first principle of data ethics. An individual has personal ownership over his information/ details. Just as it is unlawful to steal, it is unlawful to use someone's personal data without their consent. Permission should always be asked and taken before taking any information on someone. Never assume that it is alright to take personal data without consent, legal issues may arise from it. The dataset used for this study was derived from a public repository,r and contained no personal information removing any potential legal matters that may arise.

Transparency: How has the collected data been used and stored? The dataset used for this assessment was publicly available on the UCI Machine learning repository. It contains no data subject's information that can read behavious to track.

Privacy: If consent is given by a customer to collect, store and analyze personally identifiable information, does that mean they want it public? In this case, there is no such onformation hence within the context of legality and ethics, there is nothing to worry about.

Intention and Outcomes

The concept of intention matters when discussing any branch of ethics. Before data is collected, one needs to ask oneself a number of questions.
What is the data needed for?
What changes will be made after the analysis?
If the end goal is malicious- profiting from the vulnerability of others, then, it is not ethical to collect the data in the first place. If however, the intentions are good for instance, collecting data to

understand children's playgroup experience so that an app can be created to address the need of children being in playgroup; the intention behind collecting each piece of information should still be assessed. If there are data points that do not apply to the problem, there may be no need to collect such data. The whole essence is not to collect sensitive data unnecessarily. Strive to collect the minimum viable amount of data, so you're taking as little as possible from your subjects while making a difference.

For the purpose of this analysis, the intention was to make predictions using some machine learning algorithms. Will an application to a nursery school be successful or rejected, will an application to a nursery be highly recommended, prioritized or summarily dropped?

Outcomes: Even when intentions are good, the outcome of data analysis can cause inadvertent harm to individuals or groups of people. This is called a **disparate impact**, which is outlined in the Civil Rights Act as unlawful.

Conclusion

Using the decsion tree classifier to make predictions explaining the reason for rejecting a nursery school application in Slovenia provides us a 99% accuracy when compare with the KNearest Neighbors (KNN) classifier. Therefore, it Is better to use decision tree classifier than KNN to predict the outcome of an application to nursery schools in Slovenia, taking Ljubana as a sample.

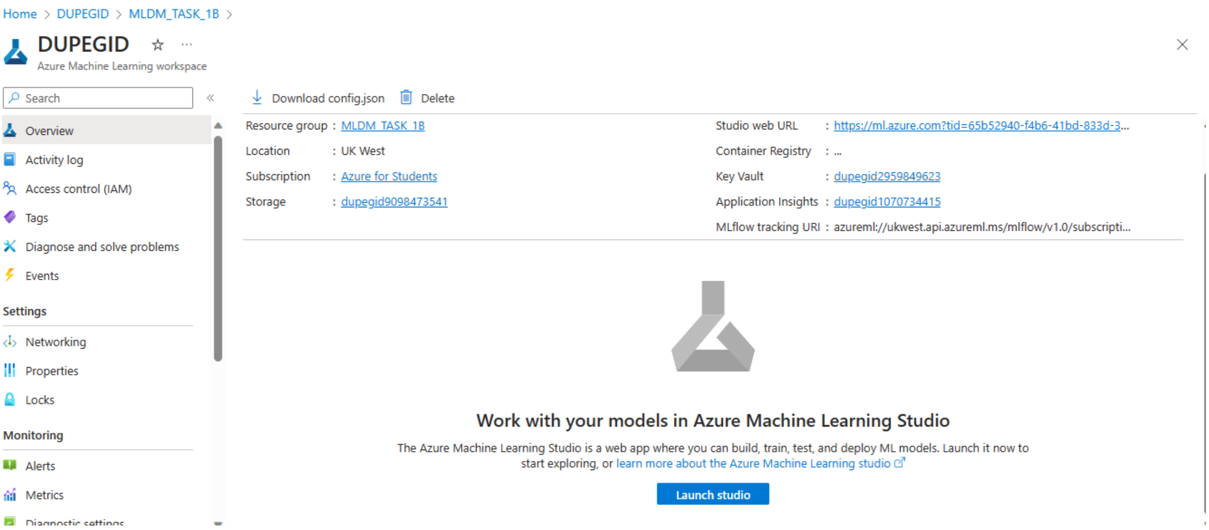
TASK 1B

USING AZURE MACHINE LEARNING

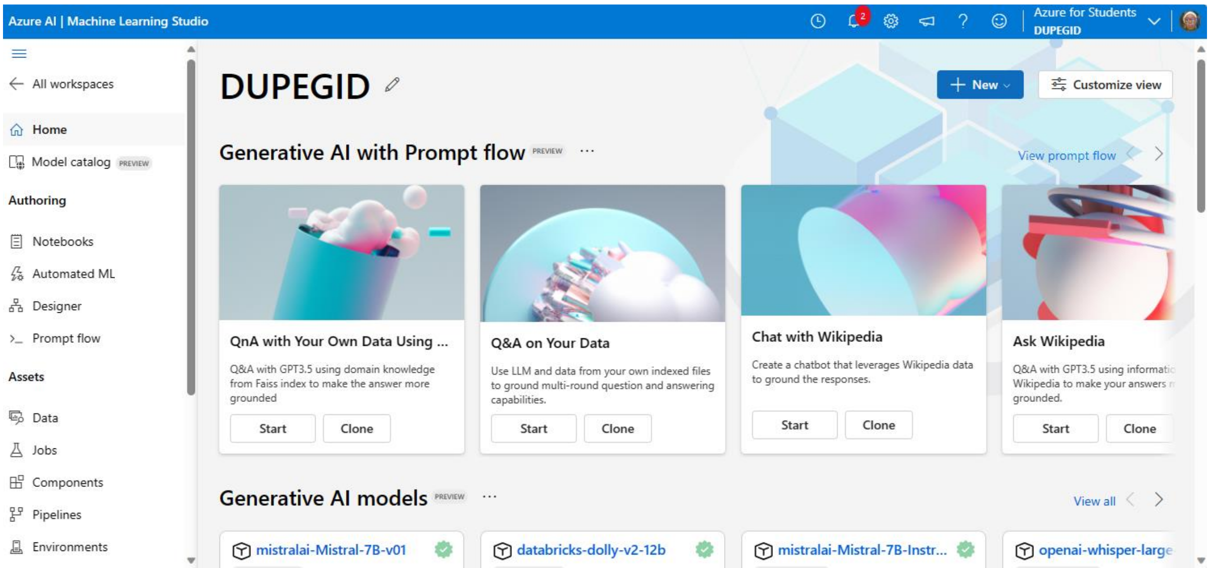
Azure Machine Learning studio allows us to do the same thing as a Jupyter notebook but in this case, it uses the drag and drop method instead of writing codes. Using azure machine learning, we are trying to make the same predictions as we did for classification. For this task, we will use the 'Nursery' Dataset as provided on the UCI machine learning repository. It has a class feature which is the target variable with labels that the model aims to predict. The labels are:
Priority
Not recom
spec_prior
very recom

Priority meaning that the application is prioritised above others, not_recom meaning that the nursery application is not recommended, spec_prior means that the application has a special priority above others and finally the very_recom means the application is highly recommended.

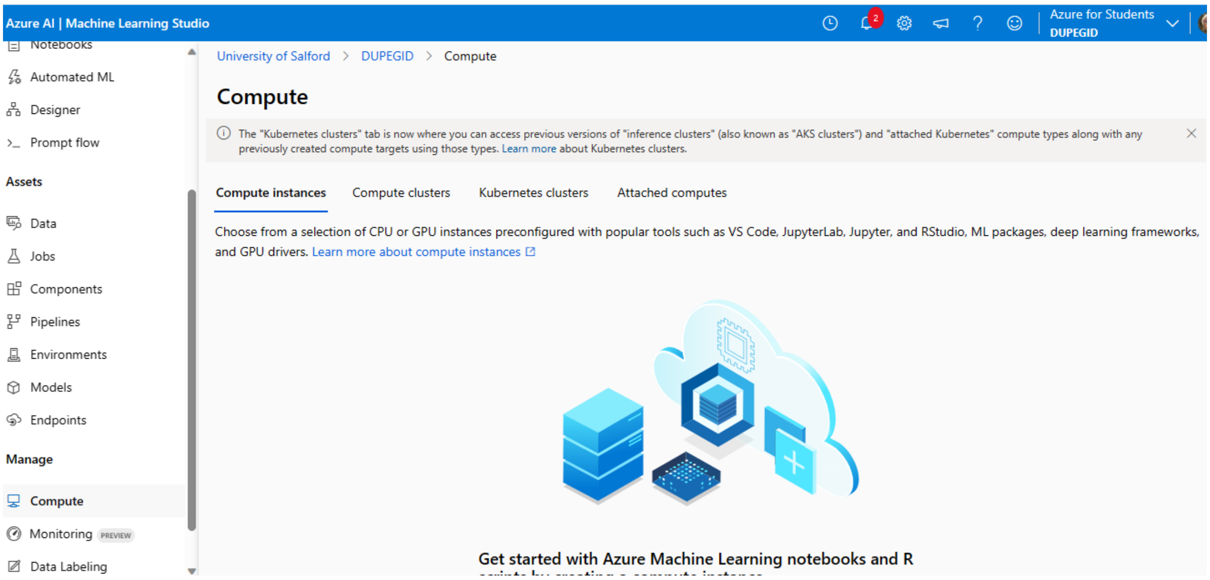
We begin by launching the Azure Machine Learning Studio



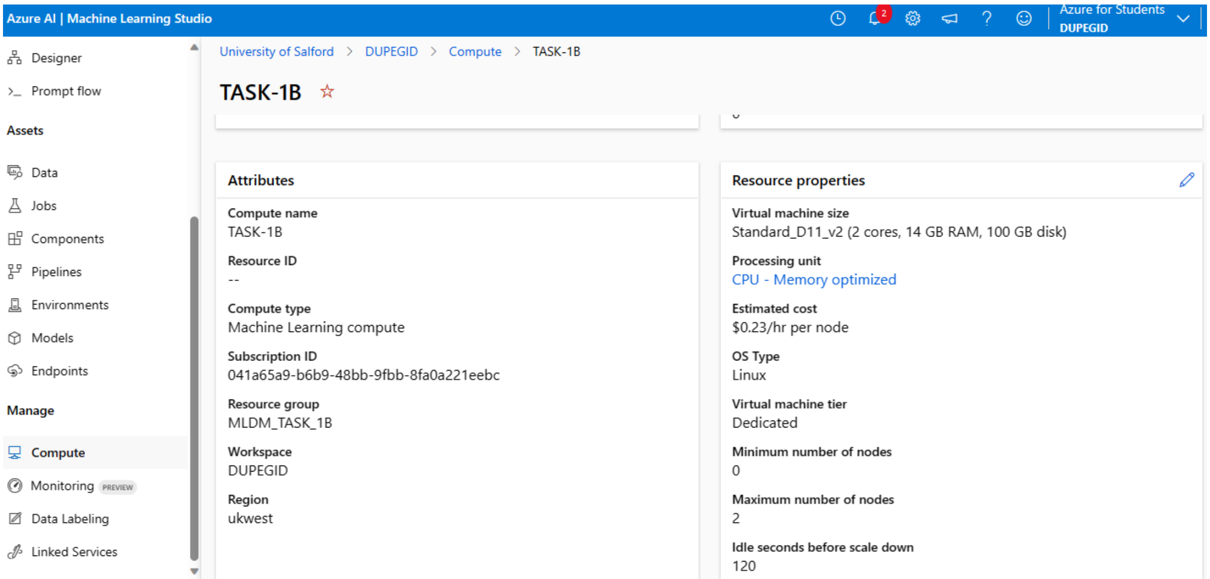
The azure machine learning studio



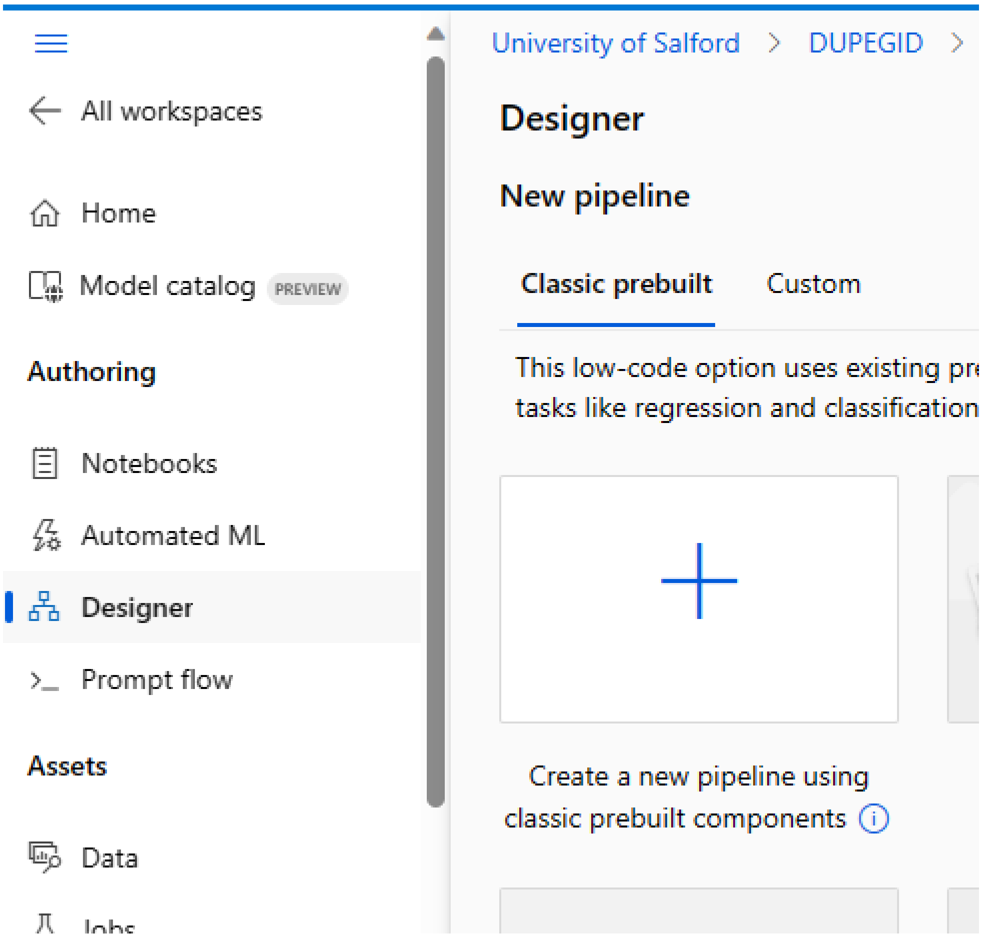
Under Manage, click Compute and under compute, select Compute Clusters x



After creating the cluster



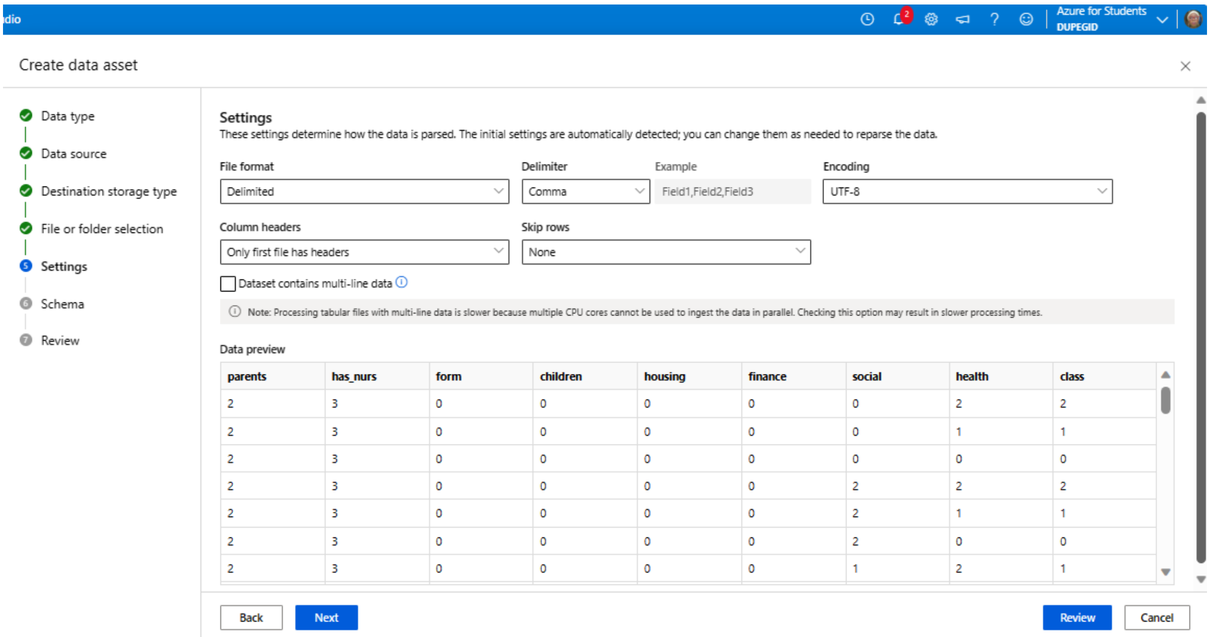
After creating the cluster, the next step is to create a pipeline in Designer. At the left pane, under All Workspaces, and under Author, view the Designer tab and select +create



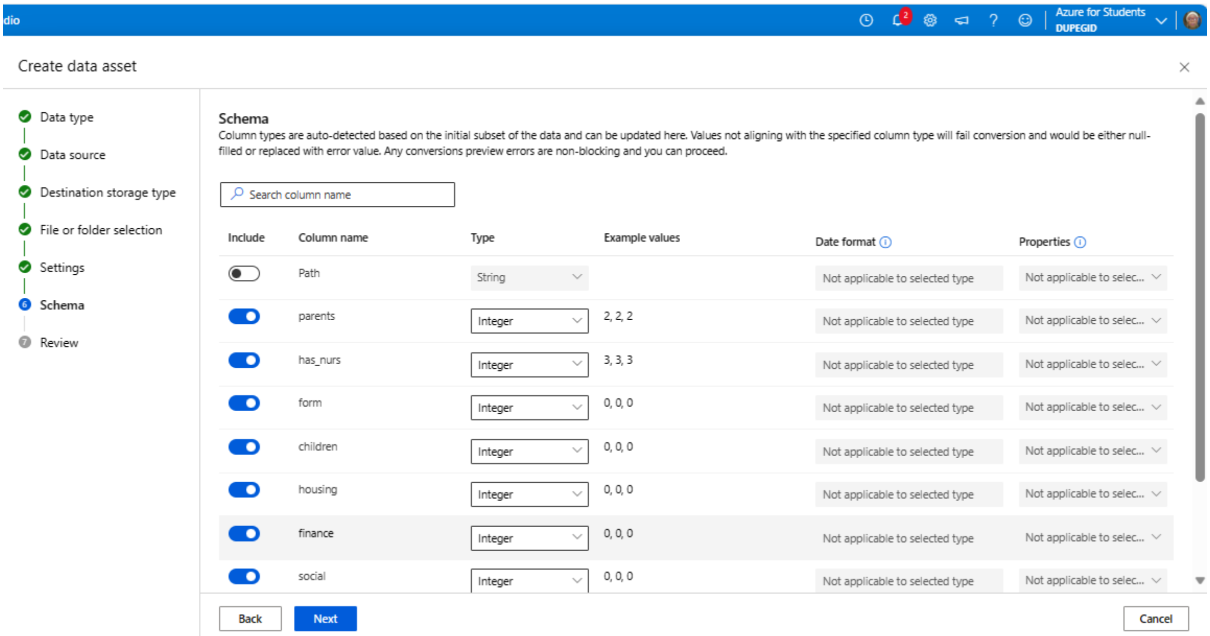
a new pipeline.

Selecting Data under Assets, use the +Create to create a dataset. In this case, the Nursery Dataset was created. To view the created dataset, it can be opened with the Explore tab as below. Click on the view profile tab to see the distribution of the columns.

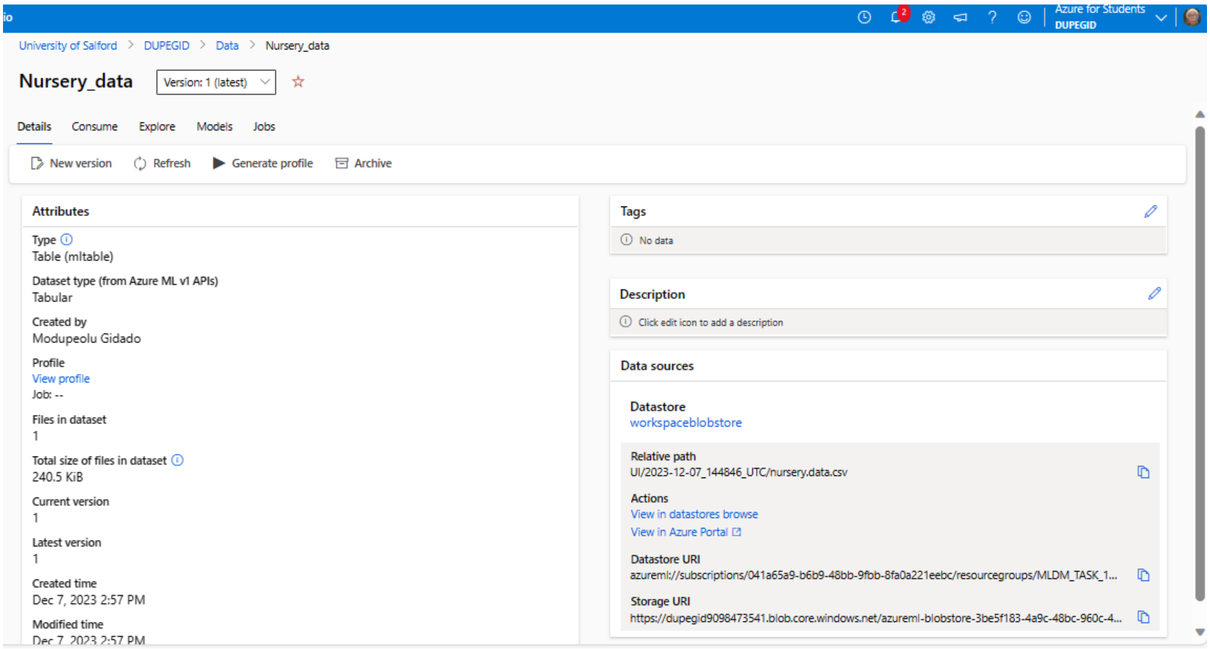
Previewing the dataset in settings. It contains 8 columns and one target variable column- class target.



Establishing the data types contained in the dataset. They are all of integer data type.

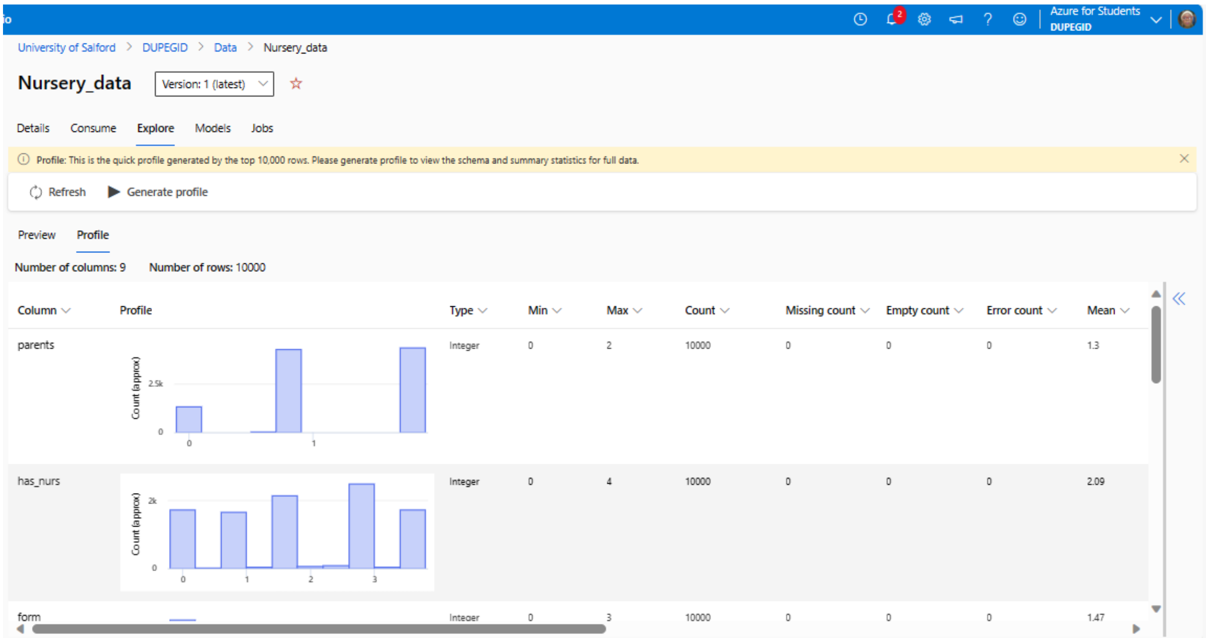


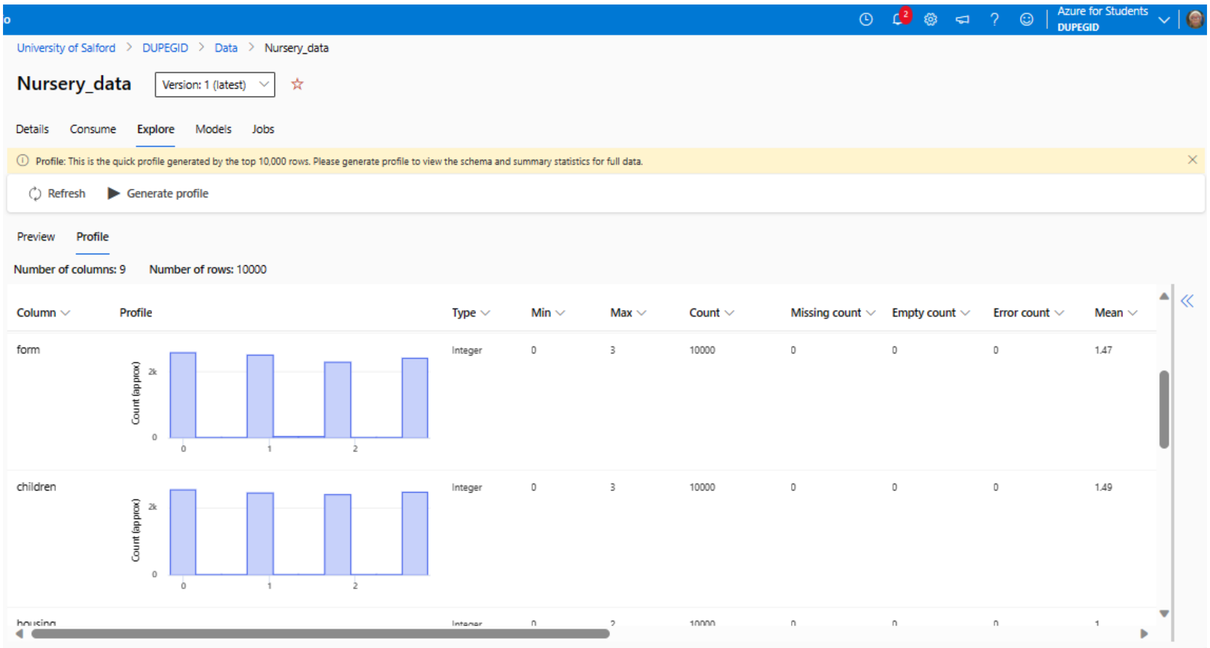
The new dataset has been successfully created as shown below.



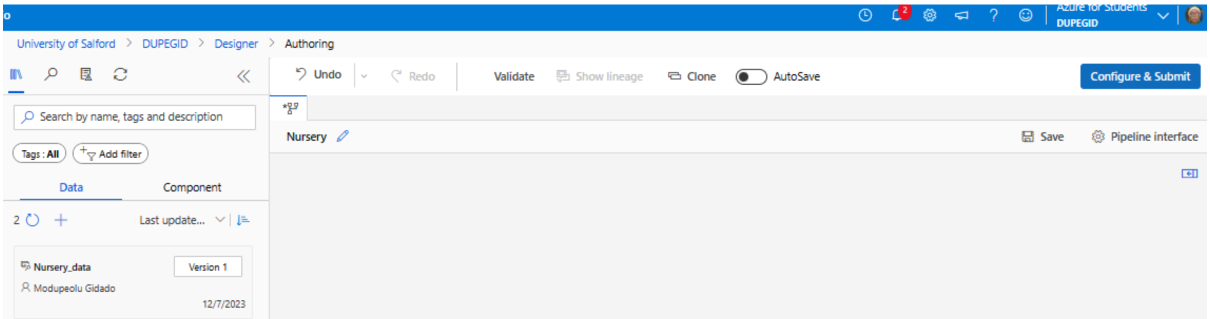
Data Exploration for the columns of the dataset

To explore the newly created dataset, we view the data distribution of the columns.





To load the data on to a canvas, we go back to the pipeline by selecting the Designer tab and on the page, we select the Nursery pipeline. Expanding the pane, we notice 2 buttons- Data and Component. Clicking on Data will present our just created data pipeline. Using the drag and drop feature, the data is loaded on to the canvas on the right-hand pane. There was no need to normalize this dataset and also perform transformations because the values are in the same scale.



Add Training Modules

Opening the Nursery Pipeline, in the Asset pane undr components, search for Split data. Load this on to the canvas right under Nursery. After loading the data onto the canvas, to evaluate the performance of the model, we split the data into training(80%) and test(20%) sets. The training set is used to train the machine learning model from the patterns and relationships within the training set. The test set is then exposed to the trained model to assess how it generalizes to new and unseen data.

The output on the far right pane is produced.

Splitting Mode:

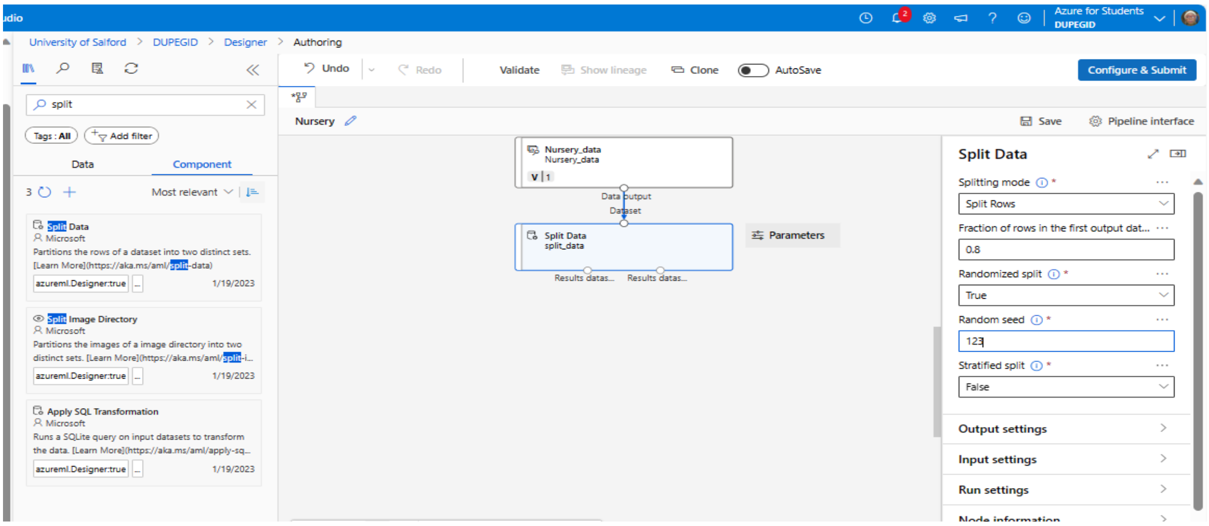
Split Rows

Fraction of rows in the first output dataset: 0.8

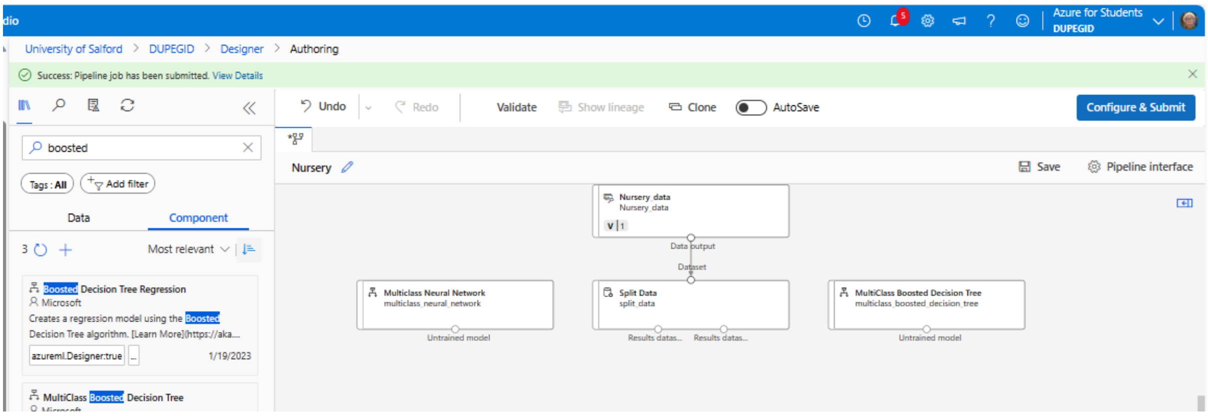
Randomized split: True

Randon Seed: 123

Stratified Split: False

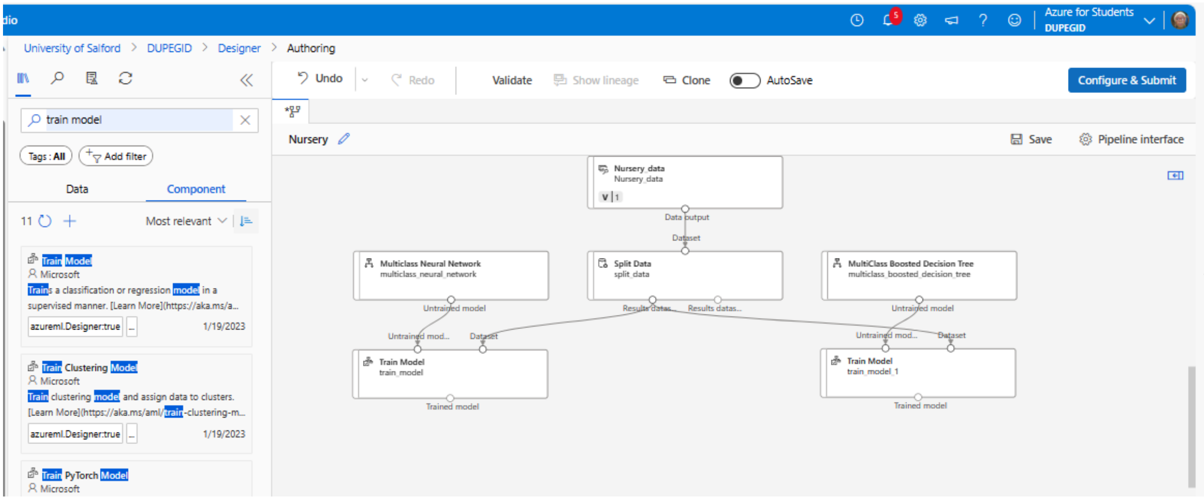


For making predictions on the dataset, the algorithms chosen are: Multiclass boosted Decision Tree and Multiclass Neural Networks because the target variable of the dataset has more than two classes. Both algorithms are run concurrently.
In the Asset Library, search for and drag Multiclass Neural Network on to the left side of the canvas, right beside Split Data. The same is done for Multiclass Boosted Decision Tree which is dragged to the right side.

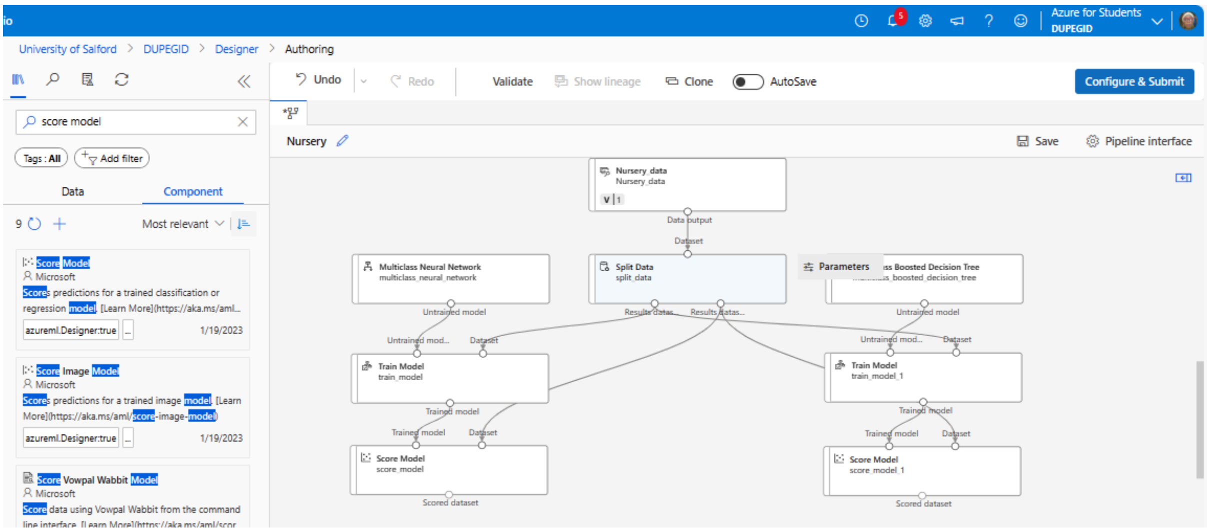


Train the models in both algorithms.
In the Asset Library, search for and drag Train Model module on to the canvas right under each

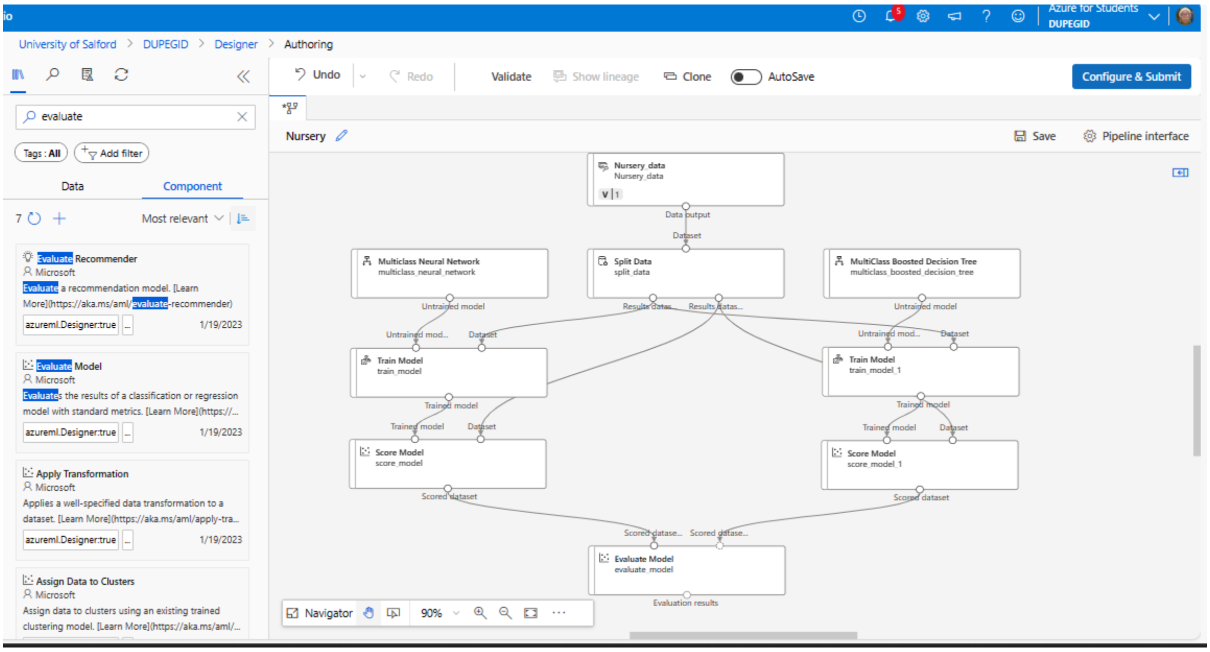
algorithm. Connect the output of the untrained model for Neural Networks to Train Model under it. Connect the output of the untrained model of the Boosted Decision Tree to Train Model under it. This will be train model 1. Another connecting line is drawn from result dataset under Split Data to Dataset in both Training models.



When using the trained models to make pedictions on the test dataset In the Asset Library, search for and drag Score Model on to the canvas. Plcae this right under the training models. Connect the train model under Train model module to that on Score Model module. Do this for both Training Models. Connect the other Result Dataset under Split data to data on Score Model module and Score Model 1.

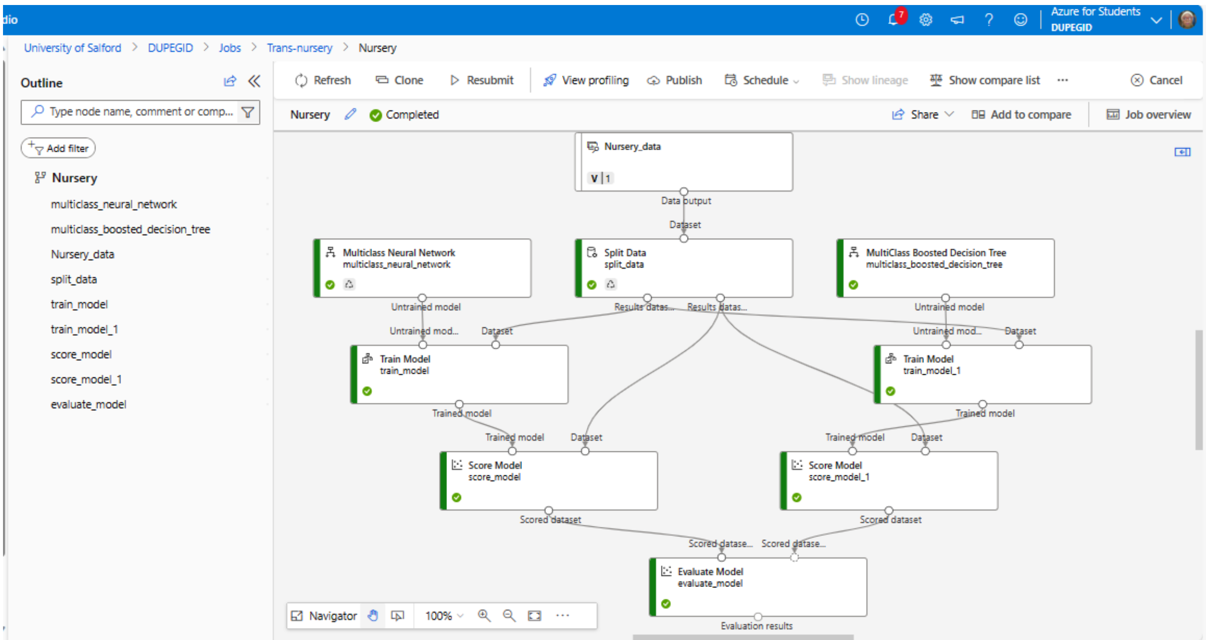


Evaluate the performance of the models on the test dataset In the Asset Library, search for Evaluate Model module, drag this on to the canvas right in the middle of both Score Model and Score Model 1. Drag a connecting line from both Score Models to the Evaluate Model Module



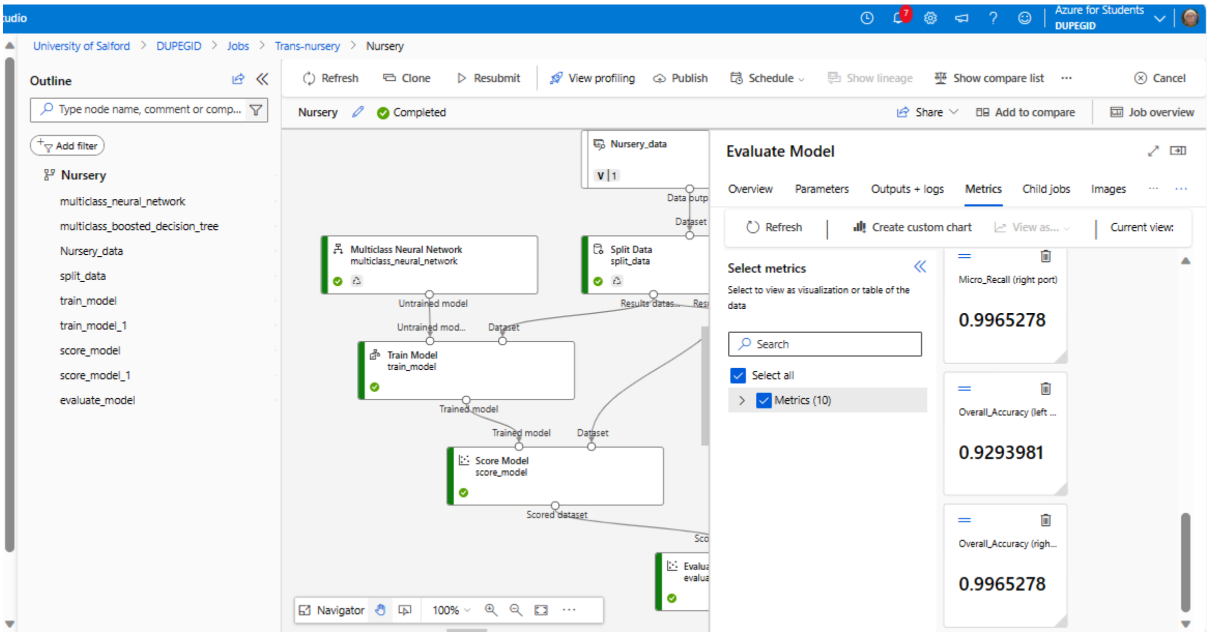
The completed pipeline

After performing all of the above, on the top righ had of the dashboard, click on Configure & Submit and then Submit to run the pipeline under the Trans-nursery experiment.



At completion, select Pipeline on the left hand panel and click on the last display name. Right-click Evaluate Model on the canvas on the new tab. Click Preview Data and select Evaluation Results to view the performance metrics. These metrics will help to assess how well the model predicts based on the test data. Evaluation results for the performance of the model. Boosted Decision Tree- 99%, Neural Network- 92%.

NOTE: There is no confusion matrix for this model evaluation.



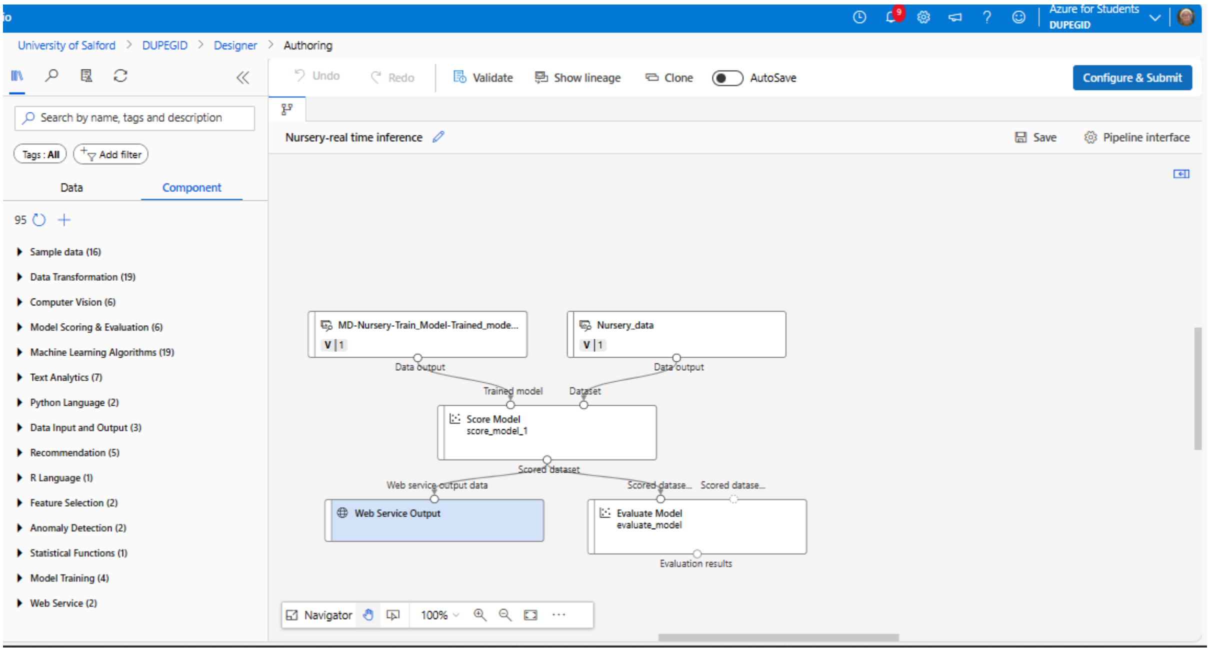
Discussion and Conclusion

From the above result, though both results are very good, based on accuracy, the boosted Decision Tree model make more accurate predictions on the dataset. It has 99% accuuracy compared with the Multiclass Neural Network that precists 92%

Inference Pipeline

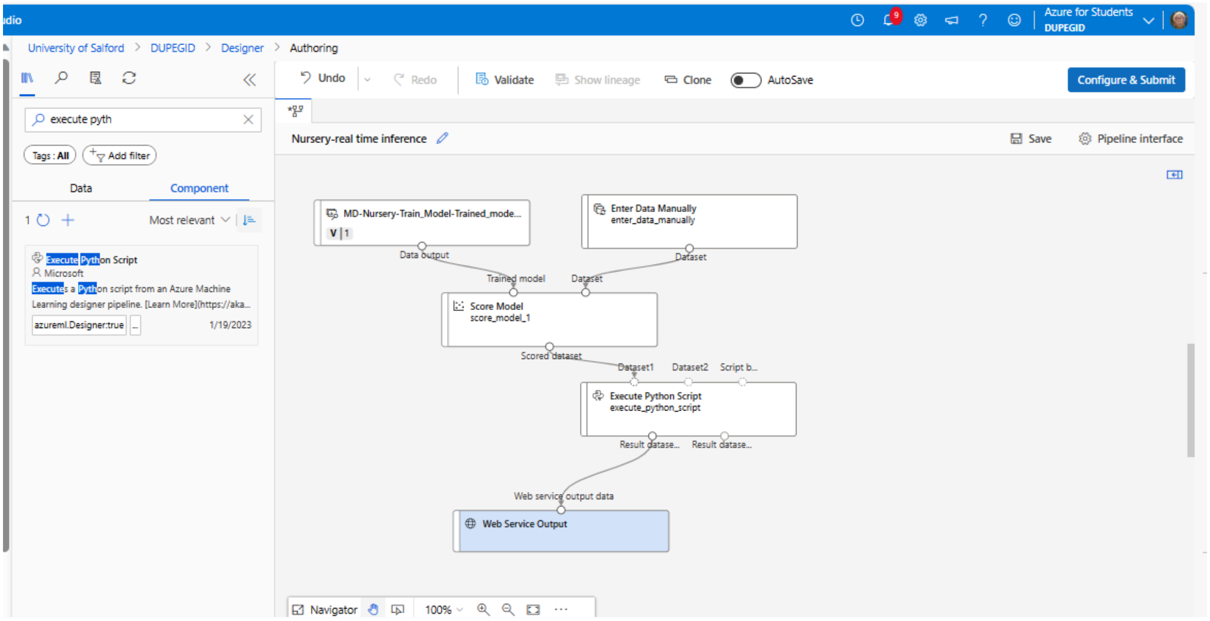
Create an inference pipeling using the Multiclass Boosted Decision Tree model

Expanding the left hand pane in Azure Machine Learning Studio, select the 3 lines at the top left of the screen. Click on Jobs to view all executed jobs and select the MD-Nursery Train Model and the Nursery data pipeline.
Click on the 3 dots at the top right corner and select Create Inference Pipeline.
In the dropdown list in Create inference Pipeline, click Real-time inference pipeline.

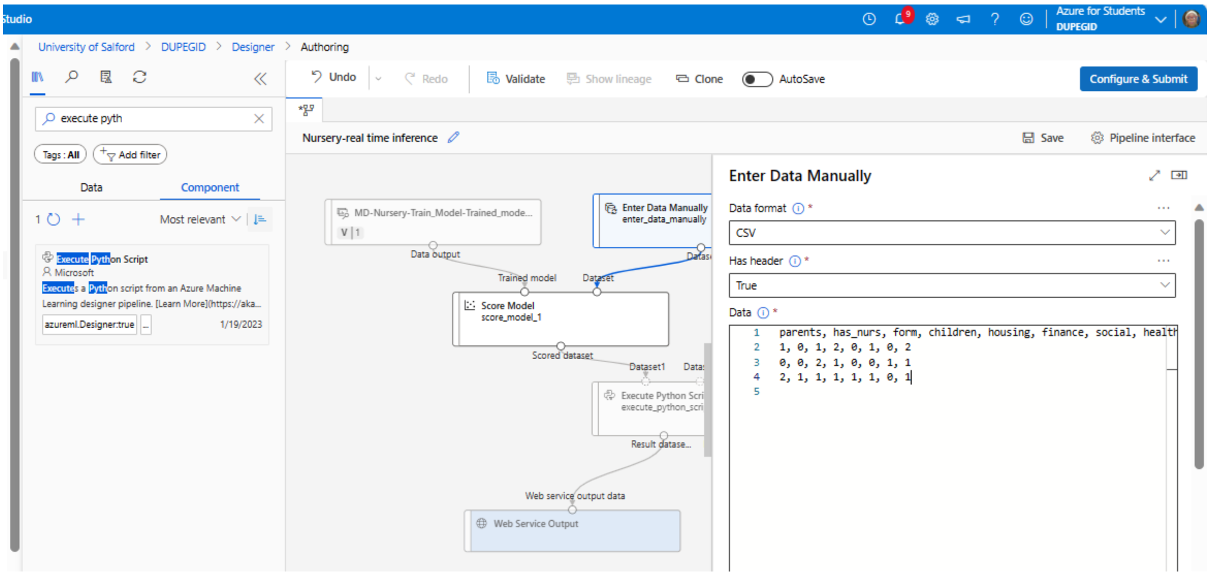


The new pipeline contains a webservice input that allows for new data to be submitted. And a web service output to return the results. The 'Nursery.data' will be removed from the above pipeline and replaced with 'Enter Data Manually', this will not include the 'class' label. It will contain a csv file with features without labels for 3new observations. While 'Evaluate Model' will be replaced with 'Execute Python Script'. Web service input is added. Connect Enter Data Manually to Dataset on Score Model_1. Connect Data Output to Trained model on Score Model1 module. Connect the Scored dataset under the Score Model module to Dataset 1 in Execute Python Script module.

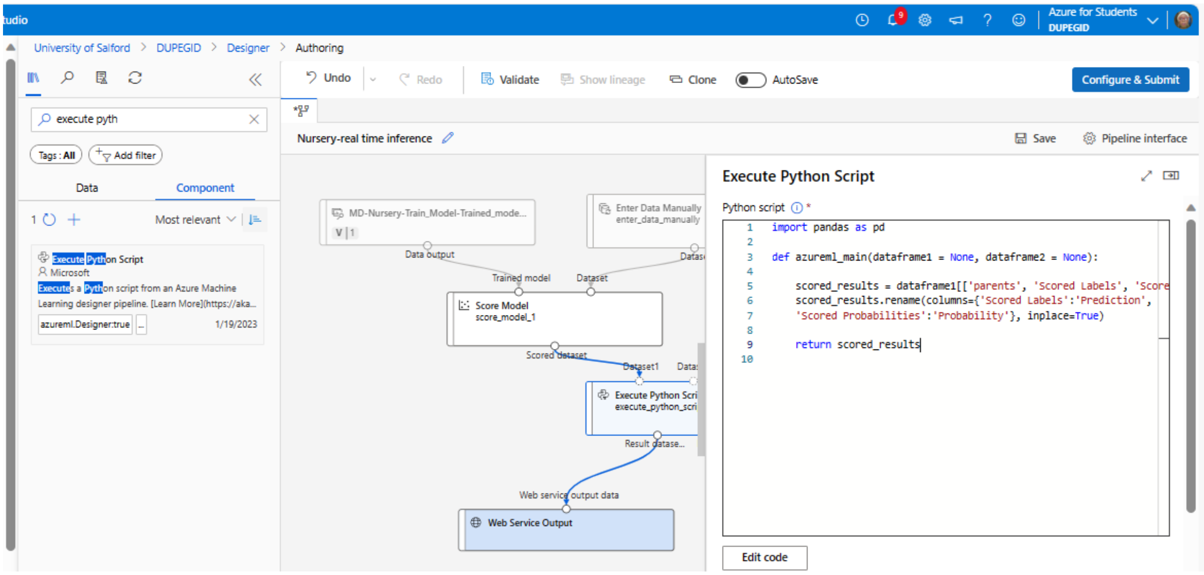
Connect the result datdaset fro Execute Python Script to the Web Service Output Data.



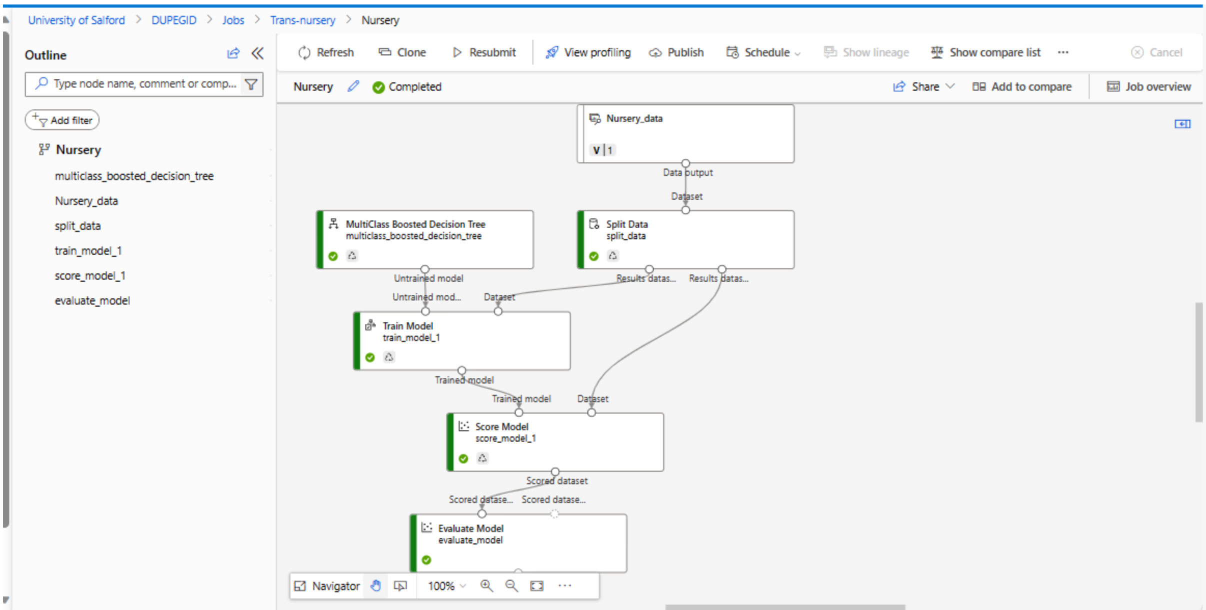
The data below was randomly formulated for the purpose of the inference pipeline.



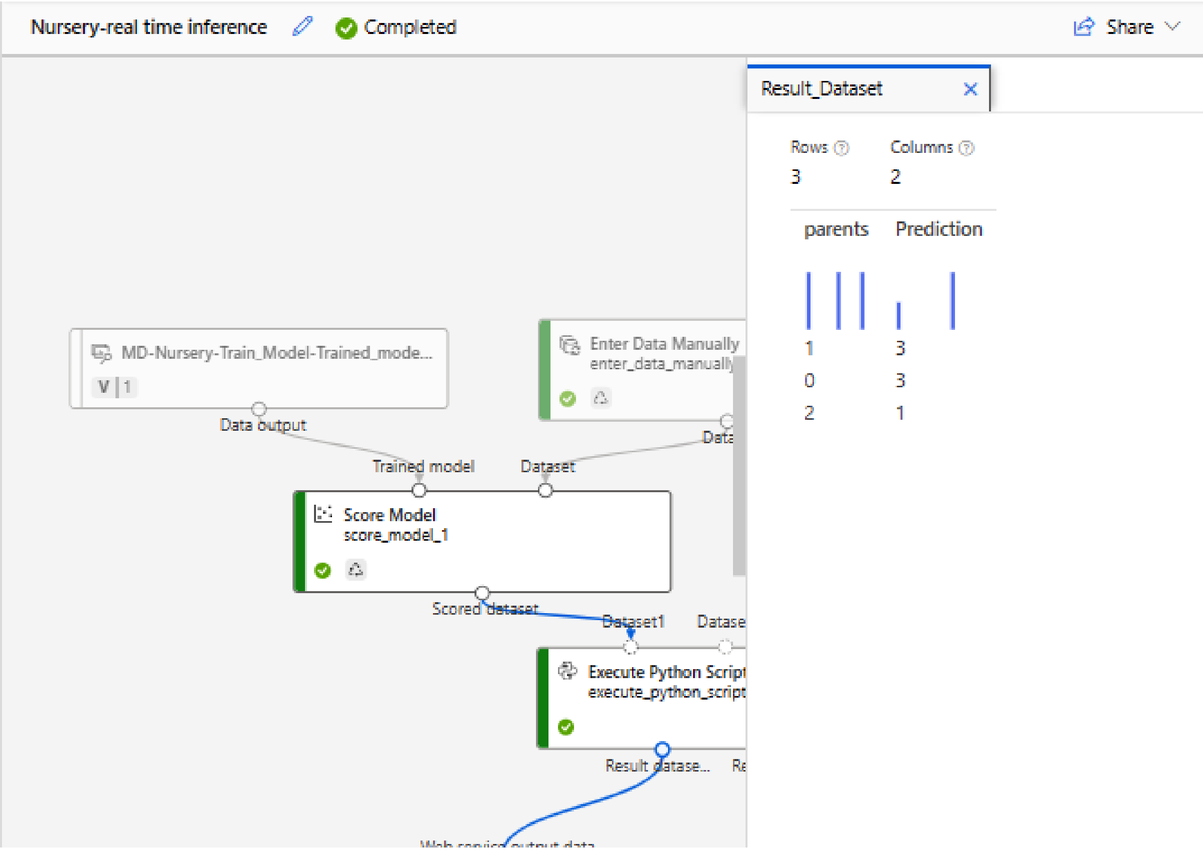
The below python code was used to return a few columns from the results of the Score Model



Inferential…



The model's prediction for the manually entered dataset



The above shows the predictions for the manually entered parents.

TASK 2: Cluster Analysis to Segment Customers Based on their Reviews

Introduction

Clustering is a form of unsupervised machine learning technique that involves grouping similar data points together. The objective of grouping the data points together is to identify the patterns in each dataset. It does not rely on labelled data for training, instead, it identifies patterns based on the inherent structure of available data. The main aim of clustering are as follows among others- to discover the structure of a dataset, group similar data points, data understanding, feature extraction and anomaly detection. For the purpose of this assessment, we will use clustering to assess the correlation between a set of reviews on destinations in 10 categories taken across East Asia.

Explanation and Preparation of Dataset

This dataset was gotten from the UCI Machine Learning Repository. It contains 10 categories of various traveller reviews on destinations across East Asia. The ratings are mapped as follows: Excellent-4, Very Good- 3, Average -2, Poor- 1, and Terrible- 0.

Below is the attribute information of the Travel reviews Dataset.

S/N	Attribute/ Features	Data Type Information
1	Unique User ID	Numeric (1-980)
2	Category 1 (Av user feedback on art galleries)	Numeric
3	Category 2 (Av user feedback on dance clubs)	Numeric
4	Category 3 (Av user feedback on juice bars)	Numeric
5	Category 4 (Av user feedback on restaurants)	Numeric
6	Category 5 (Av user feedback on museums)	Numeric
7	Category 6 (Av user feedback on resorts)	Numeric
8	Category 7 (Av user feedback on parks/ picnic spots)	Numeric
9	Category 8 (Av user feedback on beaches)	Numeric

10	Category 9 (Av user feedback on theatres)	Numeric
11	Category 10 (Av user feedback on religious institutions)	Numeric

We begin to import the libraries needed for this task.

```
In [1]:  #Importing the libraries

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

Reading the data frame- travelrev into csv

```
In [3]:  # Import the dataset

         travelrev = pd.read_csv('tripadvisor_review.csv')
```

Exploring the dataset using travelrev.datahead() to view the first and last 5 rows

```
In [5]:  #Exploration of the dataset

         travelrev.head()
```

Out[5]:

	User ID	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8	Category 9	Category 10
0	User 1	0.93	1.8	2.29	0.62	0.80	2.42	3.19	2.79	1.82	2.42
1	User 2	1.02	2.2	2.66	0.64	1.42	3.18	3.21	2.63	1.86	2.32
2	User 3	1.22	0.8	0.54	0.53	0.24	1.54	3.18	2.80	1.31	2.50
3	User 4	0.45	1.8	0.29	0.57	0.46	1.52	3.18	2.96	1.57	2.86
4	User 5	0.51	1.2	1.18	0.57	1.54	2.02	3.18	2.78	1.18	2.54

Confirming that there are no null values,

```
In [10]: # Testing if there are any missing values on the dataset
         travelrev.isnull()
```

Out[10]:

	User ID	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8	Category 9	Category 10
0	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...
975	False	False	False	False	False	False	False	False	False	False	False
976	False	False	False	False	False	False	False	False	False	False	False
977	False	False	False	False	False	False	False	False	False	False	False
978	False	False	False	False	False	False	False	False	False	False	False
979	False	False	False	False	False	False	False	False	False	False	False

980 rows × 11 columns

To get a concise summary of the structure of the dataset, showing the number of null values, datatypes, and memory usage. In this case, there are 11 columns, no null values, and datatype is float64. Memory usage is 84.3+ KB.

```
In [6]: # Getting a sneak peak into the dataset structure

        travelrev.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 980 entries, 0 to 979
Data columns (total 11 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   User ID      980 non-null    object
 1   Category 1   980 non-null    float64
 2   Category 2   980 non-null    float64
 3   Category 3   980 non-null    float64
 4   Category 4   980 non-null    float64
 5   Category 5   980 non-null    float64
 6   Category 6   980 non-null    float64
 7   Category 7   980 non-null    float64
 8   Category 8   980 non-null    float64
 9   Category 9   980 non-null    float64
 10  Category 10  980 non-null    float64
dtypes: float64(10), object(1)
memory usage: 84.3+ KB
```

Showing the statistical representation of the dataset.

In [7]:

```
# Expressing the statistical values of the dataset
travelrev.describe()
```

Out[7]:

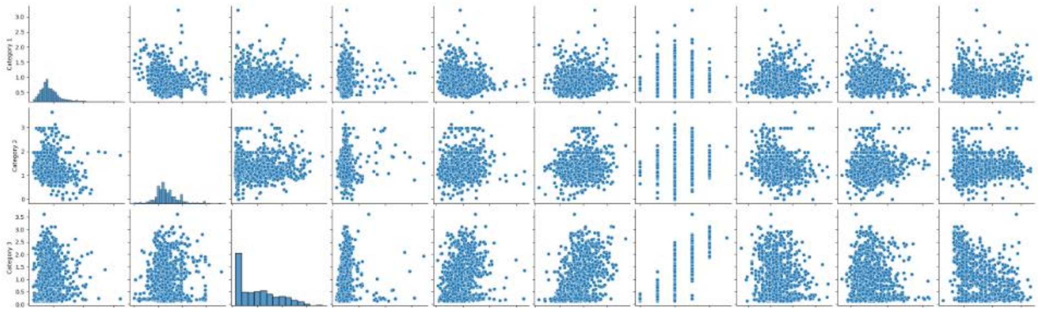
	Category 1	Category 2	Category 3	Category 4	Category 5	Category 6	Category 7	Category 8	Category 9	Category 10
count	980.000000	980.000000	980.000000	980.000000	980.000000	980.000000	980.000000	980.000000	980.000000	980.000000
mean	0.893194	1.352612	1.013306	0.532500	0.939735	1.842898	3.180939	2.835061	1.569439	2.799224
std	0.326912	0.478280	0.788607	0.279731	0.437430	0.539538	0.007824	0.137505	0.364629	0.321380
min	0.340000	0.000000	0.130000	0.150000	0.060000	0.140000	3.160000	2.420000	0.740000	2.140000
25%	0.670000	1.080000	0.270000	0.410000	0.640000	1.460000	3.180000	2.740000	1.310000	2.540000
50%	0.830000	1.280000	0.820000	0.500000	0.900000	1.800000	3.180000	2.820000	1.540000	2.780000
75%	1.020000	1.560000	1.572500	0.580000	1.200000	2.200000	3.180000	2.910000	1.760000	3.040000
max	3.220000	3.640000	3.620000	3.440000	3.300000	3.760000	3.210000	3.390000	3.170000	3.660000

Using pairplot to view the entire dataset:

In [9]:

```
#to view all the dataset using plot

sns.pairplot(travelrev)
```

Out[9]: <seaborn.axisgrid.PairGrid at 0x1d8abf2f590>



Before beginning to perform clustering on the dataset, we need to normalise selected variables (columns 5 and 11) using the StandardScaler function from sklearn.preprocessing so that some variables are not underrepresented or overlooked.

In [10]:

```
# to select the two variables that make up the travel reviews

from sklearn.preprocessing import StandardScaler
X = travelrev.iloc[:, [4,10]].values
sc_X = StandardScaler()
X = sc_X.fit_transform(X)
```

Implementation in Python (Jupyter Notebook)

For this assessment, we will employ 2 different clustering algorithms: K-Means and DBSCAN, Heirarchical

a) K-Means Clustering Algorithm

k-means clustering tries to group similar kinds of items in form of clusters. It finds the similarity between the items and groups them into the clusters. K-means clustering algorithm works in three steps: Select the k values, Initialize the centroids, Select the group, and find the average.

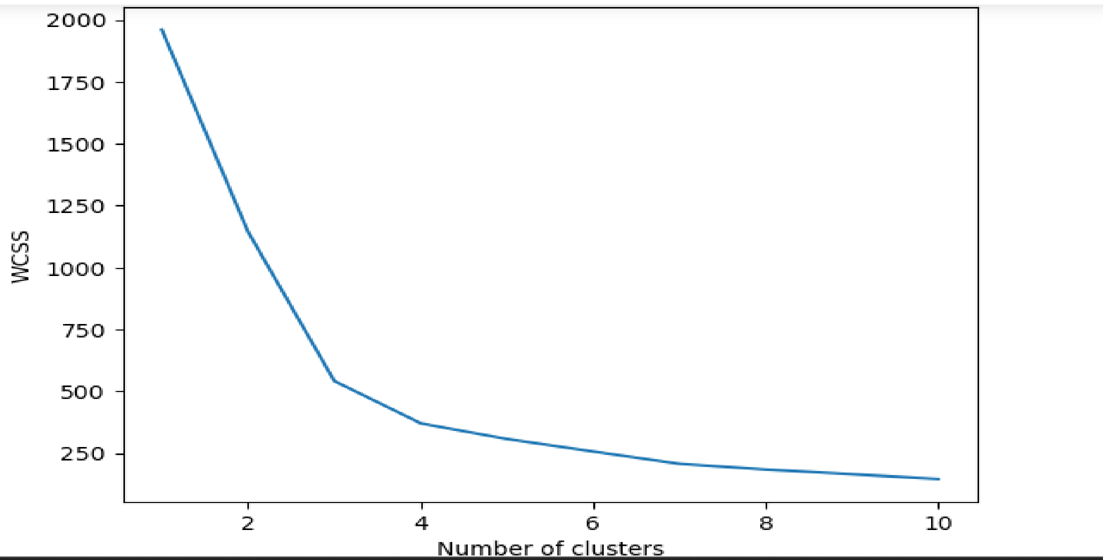
www.analyticsvidhya.com/blog/2020/10/a-simple-explanation-of-k-means-clustering/

It is a form of unsupervised learning that structures data based on each point's euclidean distance to a centre point called a centroid. These centroids are determined by the mean of all the points that belong to the same cluster which are initially selected at random by the algorithm, then iteratively adjusts them until they reach full convergence.

After the step above, we need to determine the optimal/ maximum number of clusters from the average user feedback on juice bars (category 3) and average user feedback on theatres (category 9) using elbow method. For this data with different values of k, the elbow method calculates the 'within cluster sum of squares' (wcss). The reduction of this value is the goal of K-means clustering. It helps us to visualise how the wcss value varies as the number of clusters increase.

In [11]:
```
# using the elbow method to find the optimal number of clusters
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

```
C:\Users\modup\OneDrive\Documents\Anaconda for Jupyter\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\modup\OneDrive\Documents\Anaconda for Jupyter\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\modup\OneDrive\Documents\Anaconda for Jupyter\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\modup\OneDrive\Documents\Anaconda for Jupyter\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```

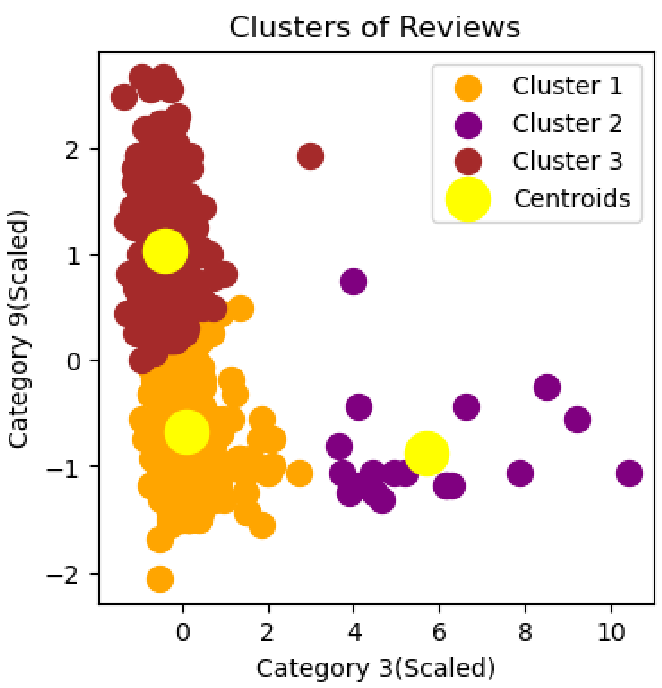


It is evident that there are 3 clusters identified. Using the syntax below, n_cluster states the number of clusters that we need the algorithm to identify.

```
In [12]:  # Using the fit_predict to train a KMeans() estimator after identifying the number of clusters
          # Fit Kmeans to the dataset

          kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
          y_kmeans = kmeans.fit_predict(X)

          C:\Users\modup\OneDrive\Documents\Anaconda for Jupyter\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The
          default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
            super()._check_params_vs_input(X, default_n_init=10)
```

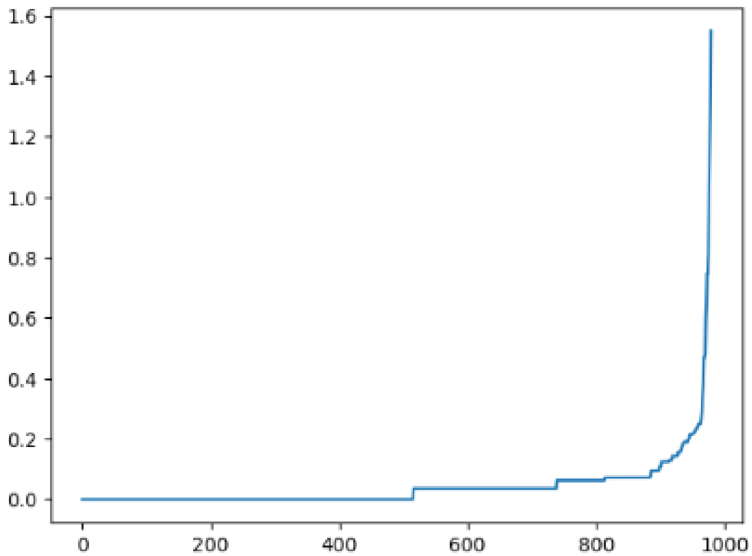
To visualize the number of clusters that have been identified, we use the syntax below.

```
In [17]:  # To visualise the clusters

          plt.figure(figsize=(4,4))
          plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'orange', label = 'Cluster 1')
          plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'purple', label = 'Cluster 2')
          plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'brown', label = 'Cluster 3')
          plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroids')
          plt.title('Clusters of Reviews')
          plt.xlabel('Category 3(Scaled)')
          plt.ylabel('Category 9(Scaled)')
          plt.legend()
          plt.show()
```



b) Using DBSCAN clustering Algorithm

```
In [18]:  # Using DBSCAN
          from sklearn.neighbors import NearestNeighbors

          neighbours = NearestNeighbors(n_neighbors=2)
          distances, indices = neighbours.fit(X).kneighbors(X)

          distances = distances[:,1]
          distances = np.sort(distances, axis = 0)
          plt.plot(distances)

Out[18]:  [<matplotlib.lines.Line2D at 0x1d8b5c954d0>]
```

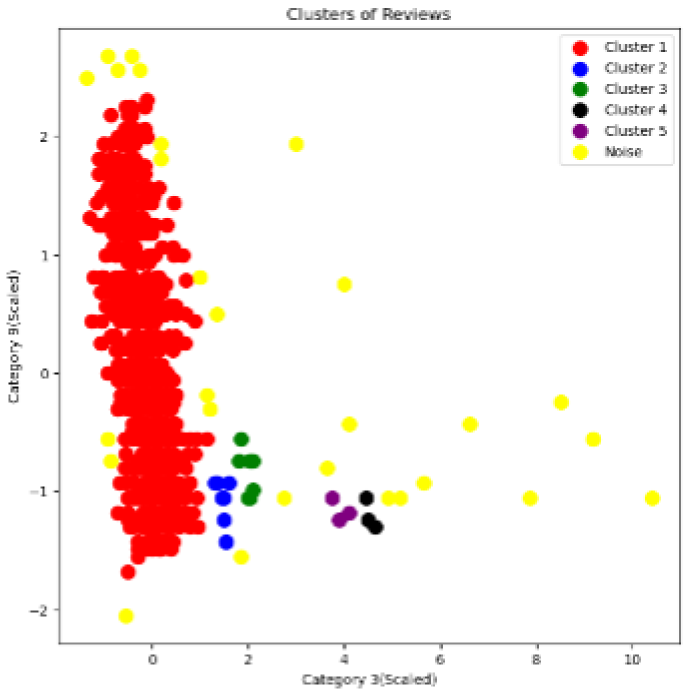


Determining the number of clusters

```
In [21]:  from sklearn.cluster import DBSCAN

          dbscan = DBSCAN(eps=0.25, min_samples = 3)
          y_dbscan = dbscan.fit_predict(X)

In [22]:  y_dbscan

Out[22]:  array([ 0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                  0,  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, -1,  0,  0,
                  0,  2,  0,  4,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,
                  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                  0,  0,  0,  0,  0,  2,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
                  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  3,  0,  0,  0,  0,
```

Plotting the DBSCAN graph

In [25]:
```
plt.figure(figsize=(8,8))
plt.scatter(X[y_dbscan == 0, 0], X[y_dbscan == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_dbscan == 1, 0], X[y_dbscan == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_dbscan == 2, 0], X[y_dbscan == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
plt.scatter(X[y_dbscan == 3, 0], X[y_dbscan == 3, 1], s = 100, c = 'black', label = 'Cluster 4')
plt.scatter(X[y_dbscan == 4, 0], X[y_dbscan == 4, 1], s = 100, c = 'purple', label = 'Cluster 5')
plt.scatter(X[y_dbscan == -1, 0], X[y_dbscan == -1, 1], s = 100, c = 'yellow', label = 'Noise')
plt.title('Clusters of Reviews')
plt.xlabel('Category 3(Scaled)')
plt.ylabel('Category 9(Scaled)')
plt.legend()
plt.show()
```



Clustering the data with higher dimensionality.

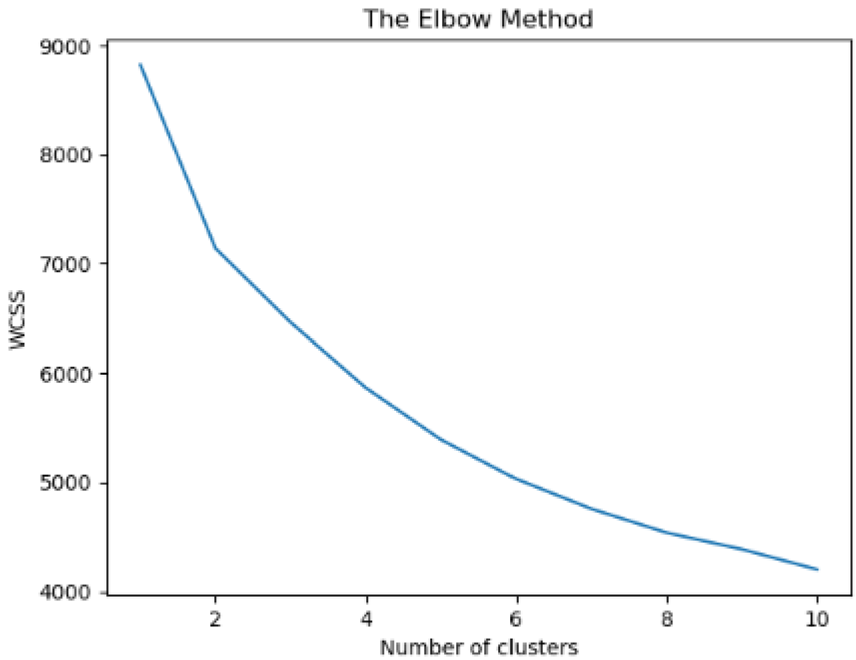
In [28]:
```
# Clustering data (Travelrev) with higher dimensionality

X = travelrev.iloc[:,1:10]

#scale the data
sc_X = StandardScaler()
X = sc_X.fit_transform(X)
```

Using the elbow method to determine the number of clusters

In [29]:
```
# employing the elbow method to determine the min no of clusters

wcss = []
for i in range(1, 11):
    kmeans1 = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans1.fit(X)
    wcss.append(kmeans1.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```

```
C:\Users\modup\OneDrive\Documents\Anaconda for Jupyter\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: Th
e default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warn
ing
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\modup\OneDrive\Documents\Anaconda for Jupyter\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: Th
e default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warn
ing
  super()._check_params_vs_input(X, default_n_init=10)
```

The elbow graph



Fitting the dataset at K2

```
In [31]:  # At k=2, fit the whole dataset

          kmeans1 = KMeans(n_clusters = 2, init = 'k-means++', random_state = 42)
          y_kmeans1 = kmeans1.fit_predict(X)

          C:\Users\modup\OneDrive\Documents\Anaconda for Jupyter\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: Th
          e default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warn
          ing
            super()._check_params_vs_input(X, default_n_init=10)
```

Result Analysis and Discussion

With K-Means clustering algorithm, 3 clusters were generated for the selected variables- categories 3 and 9 in the dataset. While using DBSCAN clustering algorithm, more clusters were generated, based on the Euclidean distance which classified other clusters as noise.

Conclusion

In conclusion, DBSCAN will be the recommended algorithm for making predictions because it is density based and can find various clusters without needing the number of clusters as an output. It was able identify more clusters indicating flexibility in cluster structures. This means that DBSAN will show reviews that are not too far apart from each other.

TASK 3: Text Mining and Sentiment Analysis for Reviews on Amazon UK Shoes

Introduction

Text mining is one of the critical ways of analysing and processing unstructured data which forms nearly 80% of the world's data. Large amounts of unstructured text data are generated daily on the internet and the process of converting these unstructured text data to a structured form so that patterns can be identified, and fresh insights obtained is what is known as text mining or data mining. It helps us to generate and establish relationships that may exist within that large amount of text data. It often uses computational algorithms to read and analyse the text information using natural Language Processing to transform the text in databases into normalized structured data good enough for analysis or machine learning algorithms.

The objective of this analysis is to determine the negative, positive and neutral reviews that customers have made on some products bought and sold on Amazon in the United States. As it should be, the impact of positive feedback in a company's growth and success cannot be overemphasized. The more positive reviews it gets, the higher the sales that will be recorded, this will impact profits, boosting the company's reputation. If contrarily, it receives negative product reviews, likes, or feedback, sales will begin to drop, reducing profits which will eventually affect the company's reputation adversely.

Explanation and Preparation of Dataset

The dataset used for this research study was gotten from dataworld, containing data on reviews of amazon UK shoe products. It has 6 columns of product_name, review_date, reviewer_name, review_title, review_text (text format) and verified_purchase. The review text column is our focus column. Below is a description of the column features and their datatypes.

S/N	Column Name	Data type
1	Product Name	text
2	Review date	text
3	Reviewer Name	text
4	Review title	text
5	Review text	text
6	Verified purchase	string

After filtering the data in an excel worksheet, the necessary libraries needed for this work was imported into the Jupyter work space and the excel read onto the pandas library.

In [47]:
```
# importing the necessary libraries

!pip install wordcloud
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import re
from wordcloud import WordCloud
import nltk
nltk.download(['stopwords',
               'punkt',
               'wordnet',
               'omw-1.4',
               'vader_lexicon'
               ])
```

Loading the data for analysis purposes using pandas. Review.head allows us see the first 5 rows.

In [48]:
```
# Reading the Data to use for the analysis

reviews = pd.read_csv('amazon-uk-shoes-product-reviews-dataset.csv')
reviews.head()
```

Out[48]:

	product_name	review_date	reviewer_name	review_title	review_text	verified_purchase
0	Klasified Women's Transparent Clear Sneaker Sh...	Reviewed in the United States on 2 June 2020	Jocelyn McSayles	Love em	Love these. Was looking for converses and thes...	True
1	Klasified Women's Transparent Clear Sneaker Sh...	Reviewed in the United States on 28 October 2021	Kenia Rivera	The plastic ripped	The shoes are very cute, but after the 2nd day...	True
2	Klasified Women's Transparent Clear Sneaker Sh...	Reviewed in the United States on 20 January 2021	Chris Souza	Good quality	Good quality	True
3	Klasified Women's Transparent Clear Sneaker Sh...	Reviewed in the United States on 22 April 2021	Amazon Customer	Good	Great	True

To view the general summary of the reviews dataframe,

In [49]:
```
#To get a general summary of the dataframe

reviews.describe()
```

Out[49]:

	product_name	review_date	reviewer_name	review_title	review_text	verified_purchase
count	3242	3242	3242	3241	3242	3242
unique	592	1335	2774	2711	3133	2
top	MEBIKE Women Cycling Shoes Lady Road Bike Shoe...	Reviewed in the United States on 3 May 2021	Amazon Customer	Five Stars	Great	True
freq	10	11	218	49	4	3230

After viewing the general summary of the dataframe, the next step is to determine what the reviews are- negative, positive, or neutral using the sentiment analyser while retaining the columns explaining the review the within the dataframe.

```
In [50]: # to view the polarity scores of the reviews while creating new columns within dataframe

         from nltk.sentiment.vader import SentimentIntensityAnalyzer

         sentiment = SentimentIntensityAnalyzer()

         reviews['compound'] = [sentiment.polarity_scores(review)['compound'] for review in reviews['review_text']]
         reviews['neg'] = [sentiment.polarity_scores(review)['neg'] for review in reviews['review_text']]
         reviews['neu'] = [sentiment.polarity_scores(review)['neu'] for review in reviews['review_text']]
         reviews['pos'] = [sentiment.polarity_scores(review)['pos'] for review in reviews['review_text']]
```

```
In [51]: reviews.head()
```

Out[51]:

	product_name	review_date	reviewer_name	review_title	review_text	verified_purchase	compound	neg	neu	pos
0	Klasified Women's Transparent Clear Sneaker Sh...	Reviewed in the United States on 2 June 2020	Jocelyn McSayles	Love em	Love these. Was looking for converses and thes...	True	0.8517	0.034	0.710	0.256
1	Klasified Women's Transparent Clear Sneaker Sh...	Reviewed in the United States on 28 October 2021	Kenia Rivera	The plastic ripped	The shoes are very cute, but after the 2nd day...	True	0.6593	0.000	0.926	0.074
2	Klasified Women's Transparent Clear Sneaker Sh...	Reviewed in the United States on 20 January 2021	Chris Souza	Good quality	Good quality	True	0.4404	0.000	0.256	0.744
3	Klasified Women's Transparent Clear Sneaker Sh...	Reviewed in the United States on 22 April 2021	Amazon Customer	Good	Great	True	0.6249	0.000	0.000	1.000
4	Aravon Women's Betty-AR Oxfords, Stone, 5.5 UK	Reviewed in the United States on 27 August 2020	Burger Lover - Dalton	Great quality and comfort shoes	Great quality and comfort shoesl was so thrill...	True	0.9949	0.021	0.695	0.284

To get a general summary of the newly created columns indicating the type of reviews.

```
In [52]: reviews[['compound','neg','neu','pos']].describe()
```

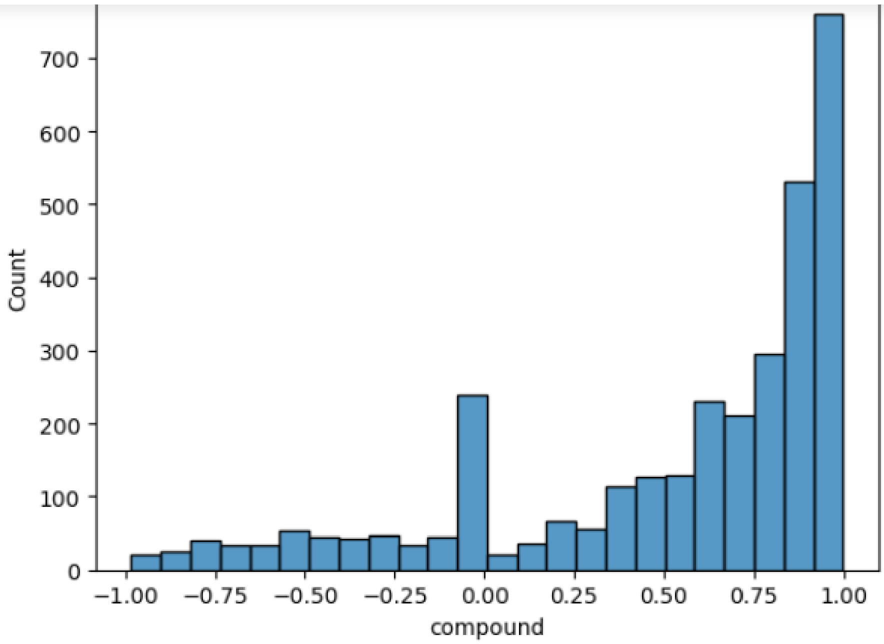
Out[52]:

	compound	neg	neu	pos
count	3242.000000	3242.000000	3242.000000	3242.000000
mean	0.531287	0.044099	0.696755	0.259137
std	0.493333	0.077361	0.202788	0.216755
min	-0.983600	0.000000	0.000000	0.000000
25%	0.294250	0.000000	0.608000	0.109000
50%	0.735100	0.000000	0.742000	0.205000
75%	0.908775	0.067000	0.828000	0.353750
max	0.998300	1.000000	1.000000	1.000000

Within a scale of -1 to 1, after compounding all the reviews, it is taken that scores between 0 and -1 are negative feedback and those between 0 and 1 are positive feedback while those at 0 represent the neutral feedback. From the description above, the compound score of most products at the median is 50% represented by 0.73 of all the reviews which means a positive sentiment to most products.

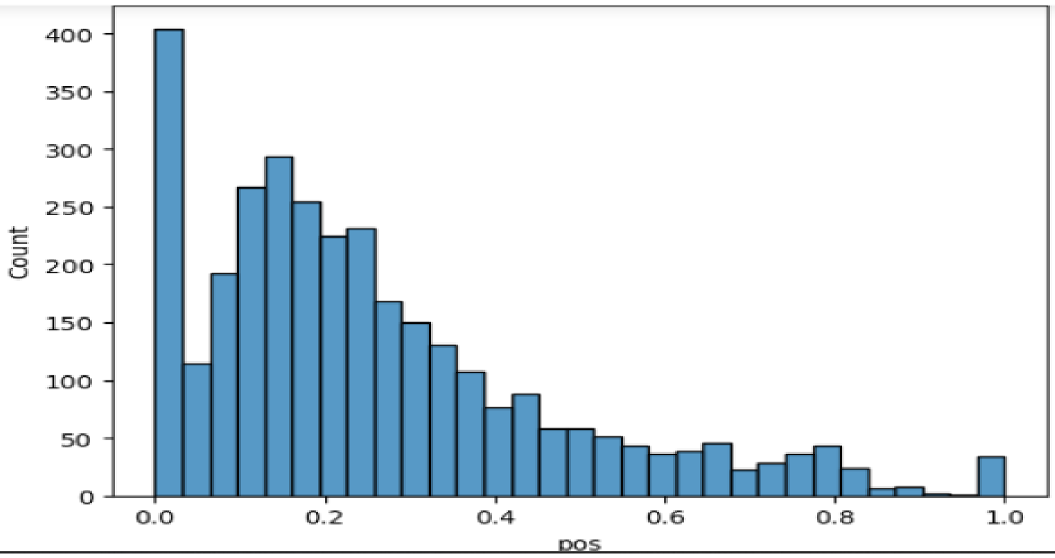
We can have a look at the distribution of the compound scores.

```
In [53]: sns.histplot(reviews['compound'])
Out[53]: <Axes: xlabel='compound', ylabel='Count'>
```



Taking a view of the distribution of the positive reviews shows most reviews are positive falling within the band of 0 and 1.

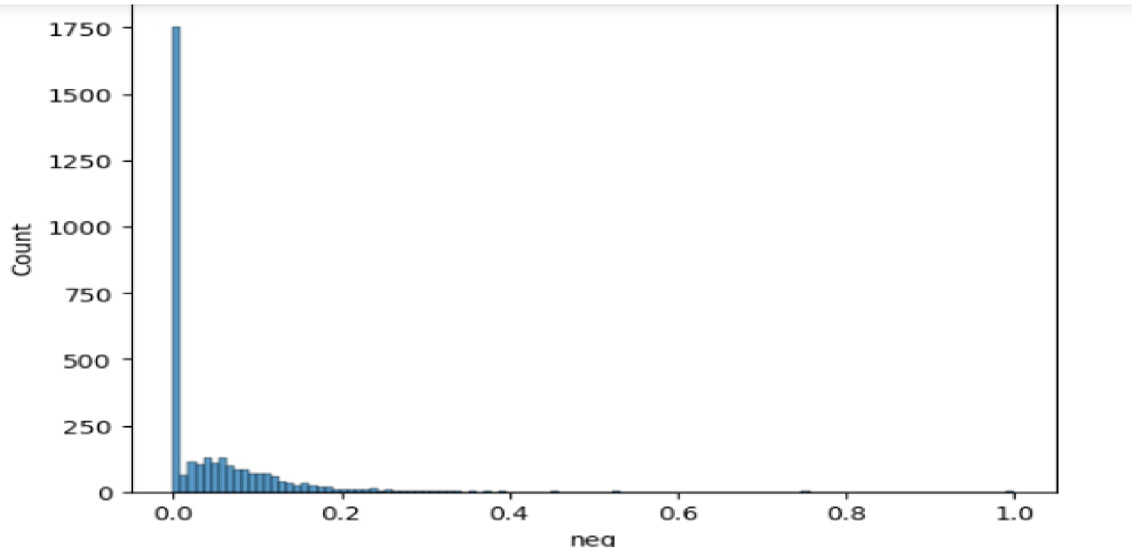
```
In [54]: # Viewing a distribution of the positive reviews
         sns.histplot(reviews['pos'])
Out[54]: <Axes: xlabel='pos', ylabel='Count'>
```



Taking a view of the negative distribution of reviews, reveals less of them are negative with 1750 neutral reviews.

```
In [55]: # a negative distribution of reviews
         sns.histplot(reviews['neg'])
```

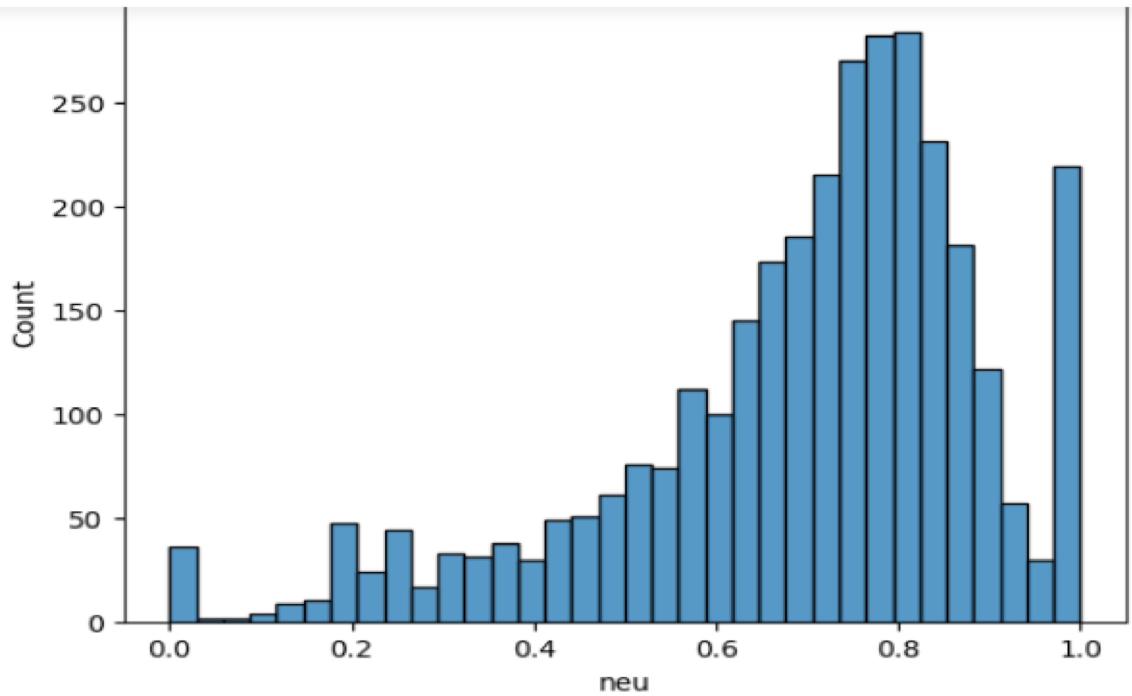
```
Out[55]: <Axes: xlabel='neg', ylabel='Count'>
```



Viewing the distribution of neutral reviews on the products reveals

```
In [56]: # viewing the distribution of neutral reviews
         sns.histplot(reviews['neu'])
```

```
Out[56]: <Axes: xlabel='neu', ylabel='Count'>
```



Checking the number of negative reviews per product,

In [57]:
```
# checking how many negative reviews per product
(reviews['compound']<=0).groupby(reviews['product_name']).sum()
```
Out[57]:
```
product_name
': 'ZAPATILLA NEW BALANCE KV220, Navy/White, 9.5 UK Child                                                    2
ALLY UNION MAKE FORCE Mens Womens Walking Shoes Lightweight Mesh Slip-on Running Sneakers Gray Size: 8 Women/7 Men    2
ANNE KLEIN Women's Anne Kleon Onthego Sneaker, Navy, 7.5 UK                                                   0
ANNE KLEIN Women's Terri Sneaker, Grey Heathered, 4 UK                                                        4
APEX LEGENDS Women's Breeze Athletic Knit Running Shoe Sneaker, Grey, 7.5 UK                                  0
                                                                                                             ..
adidas unisex child Terrex Hyperhiker Low Hiking Shoe, Grey/Black/Grey, 11 Little Kid US                      0
adidas unisex-child Duramo SL,Black/Black/Grey,10.5 M US                                                      1
adidas unisex-child Racer TR 2.0,Ink/Copper/pink tint,12.5 M US                                               2
bebe Girls' Sandals - Rhinestone Studded Criss Cross Strap Sandals (Toddler/Girls) brown Size: 6 Toddler      2
konhill Women's Walking Tennis Shoes - Lightweight Athletic Casual Gym Slip on Sneakers 8636 Mauve Size: 9.5  1
Name: compound, Length: 592, dtype: int64
```

Doing the same for positive reviews per product

In [84]:
```
# checking how many positive reviews per product
(reviews['compound']>=0).groupby(reviews['product_name']).sum()
```
Out[84]:
```
product_name
': 'ZAPATILLA NEW BALANCE KV220, Navy/White, 9.5 UK Child                                                    6
ALLY UNION MAKE FORCE Mens Womens Walking Shoes Lightweight Mesh Slip-on Running Sneakers Gray Size: 8 Women/7 Men    8
ANNE KLEIN Women's Anne Kleon Onthego Sneaker, Navy, 7.5 UK                                                   1
ANNE KLEIN Women's Terri Sneaker, Grey Heathered, 4 UK                                                        6
APEX LEGENDS Women's Breeze Athletic Knit Running Shoe Sneaker, Grey, 7.5 UK                                  2
                                                                                                             ..
adidas unisex child Terrex Hyperhiker Low Hiking Shoe, Grey/Black/Grey, 11 Little Kid US                      4
adidas unisex-child Duramo SL,Black/Black/Grey,10.5 M US                                                      9
adidas unisex-child Racer TR 2.0,Ink/Copper/pink tint,12.5 M US                                               6
bebe Girls' Sandals - Rhinestone Studded Criss Cross Strap Sandals (Toddler/Girls) brown Size: 6 Toddler      8
konhill Women's Walking Tennis Shoes - Lightweight Athletic Casual Gym Slip on Sneakers 8636 Mauve Size: 9.5  9
Name: compound, Length: 592, dtype: int64
```

Calculating the percentage of negative reviews per product.

In [58]:
```
# calculate percentage of negative reviews per product

percent_negative = pd.DataFrame((reviews['compound']<=0).groupby(reviews['product_name']).sum()
                                /reviews['product_name'].groupby(reviews['product_name']).count()*100,
                                columns=['% negative reviews']).sort_values(by='% negative reviews')

percent_negative
```

Below is the output of the percentage of negative reviews

Out[58]:

	% negative reviews
product_name	
Klasified Women's Transparent Clear Sneaker Shoe, White, 5.5 UK	0.0
adidas Kids' Adissage	0.0
Fergie Women's Shortly Slip-ons Loafer, Blue Tye Dye, 4.5 UK	0.0
OshKosh B'Gosh Unisex-Child Bia Bump Toe Mary Jane Flat pink Size: 4 Toddler	0.0
adidas Daily 3.0 Skate Shoe, Grey, 3 US Unisex Little Kid	0.0
...	...
DKNY Women's Lightweight Slip on Fashion Sneaker, Military Green Abbi, 6.5	100.0
Creative Recreation Women's Cesario Lo XVI Classic Sneaker, White, 10	100.0
Converse Chuck Taylor All Star Low Top	100.0
Kenneth Cole New York Women's KAM 10 Low TOP LACE UP Sneaker Embroidered, Light Gold, 4 UK	100.0
Kamik Boy's Unisex Kids Cassia NF8080 Snow Boot, Grey BLK, 10.5 UK Child	100.0

592 rows × 1 columns

To view the percentage of negatively reviewed products between 0 and 20%

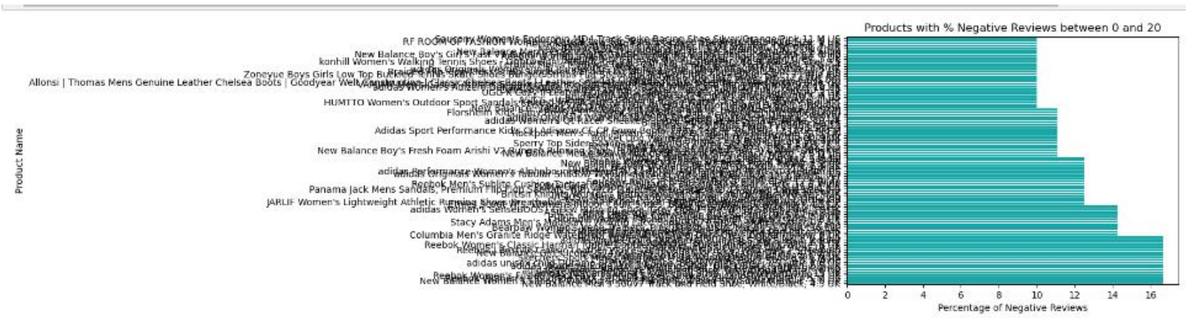
In [74]:
```
# To view the percentage negative reviewed products between 0 and 20

filtered_percent_negative = percent_negative[(percent_negative['% negative reviews'] > 0) & (percent_negative['% negative rev

print(filtered_percent_negative)
```

```
                                                    % negative reviews
product_name
Saucony Women's Endorphin MD4 Track Spike Racin...           10.000000
RF ROOM OF FASHION Women's Casual Low Top Trend...           10.000000
Reebok Women's Sole Fury TS Cross Trainer, Whit...           10.000000
Sperry Women's Crest Striper II CVO Sneaker, Oa...           10.000000
New Balance 519v1 Running Shoe, Black/Rainbow, ...           10.000000
...                                                                ...
adidas Women's Ligra 6 Volleyball Shoe, White/W...           16.666667
Reebok Women's Fusion Flexweave Running Shoe, S...           16.666667
Reebok Women's Cloudride DMX 4.0 Walking Shoe, ...           16.666667
New Balance Women's 1365v1 Walking Shoe, Chambr...           16.666667
New Balance Men's 500v7 Track and Field Shoe, W...           16.666667

[97 rows x 1 columns]
```

Using Seaborn to plot the graphical representation of this information- 0-20 percentage of negative reviews.

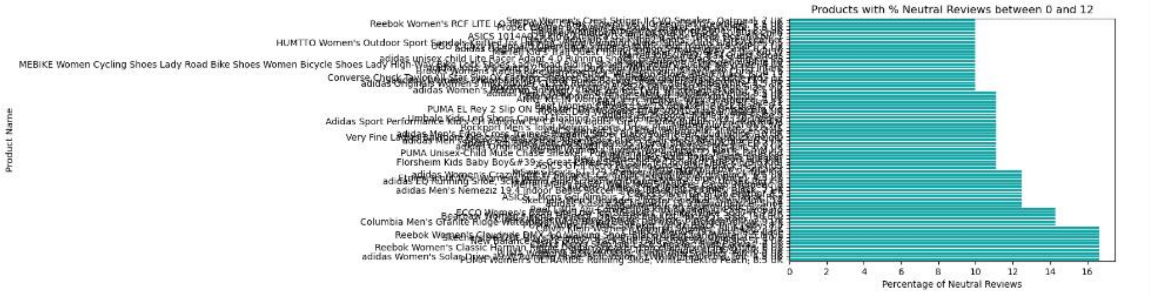
In [75]:
```
# using seaborn,this information can be plotted on a graph horizontally

# Assuming 'percent_negative' is your DataFrame
filtered_percent_negative = percent_negative[(percent_negative['% negative reviews'] > 0) & (percent_negative['% negative rev

# Plotting the horizontal barplot
sns.barplot(x='% negative reviews', y=filtered_percent_negative.index, data=filtered_percent_negative, color='c')

# Adding labels and title
plt.xlabel('Percentage of Negative Reviews')
plt.ylabel('Product Name')
plt.title('Products with % Negative Reviews between 0 and 20')

# Display the plot
plt.show()
```

With percent_negative as the data frame, plot the horizontal bar plot using 'Negative reviews' as the x-axis and product name on the y-axis, below is the output.



For neutral reviews with comments between 0 and 12 %

In [92]:
```python
# using seaborn,this information can be plotted on a graph horizontally

# Assuming 'percent_negative' is your DataFrame
filtered_percent_neutral = percent_neutral[(percent_neutral['% neutral reviews'] > 0) & (percent_neutral['% neutral reviews']

# Plotting the horizontal barplot
sns.barplot(x='% neutral reviews', y=filtered_percent_neutral.index, data=filtered_percent_neutral, color='c')

# Adding labels and title
plt.xlabel('Percentage of Neutral Reviews')
plt.ylabel('Product Name')
plt.title('Products with % Neutral Reviews between 0 and 12')

# Display the plot
plt.show()
```



Next, a function is created that applies all the preprocessing steps so that we can use them on a corpus.

In [27]:
```python
# creating a function that applies all the data preprocessing steps that we can then use on a corpus

stop_words = nltk.corpus.stopwords.words('english')

def preprocess_text(text):
    tokenized_document = nltk.tokenize.RegexpTokenizer('[a-zA-Z0-9\']+').tokenize(text) #Tokenize
    cleaned_tokens = [word.lower() for word in tokenized_document if word.lower() not in stop_words] # Remove
    stemmed_text = [nltk.stem.PorterStemmer().stem(word) for word in cleaned_tokens] #Stemming
    return stemmed_text

stop_words = nltk.corpus.stopwords.words('english')
reviews['processed_review'] = reviews['review_text'].apply(preprocess_text)

reviews_positive_subset = reviews.loc[(reviews['product_name']=='Reebok Women\'s Fusion Flexweave Running Shoe, Smokey Volcar
                                  & (reviews['compound']>0),:]

reviews_negative_subset = reviews.loc[(reviews['product_name']=='Reebok Women\'s Fusion Flexweave Running Shoe, Smokey Volcar
                                  & (reviews['compound']>0),:]

reviews_positive_subset.head()
```

Out[27]:

	product_name	review_date	reviewer_name	review_title	review_text	verified_purchase	compound	neg	neu	pos	processed_review
695	Reebok Women's Fusion Flexweave Running Shoe, ...	Reviewed in the United States on 28 October 2021	Angels best	Reebok I love u	Best gym shoe for walking and standing for lon...	True	0.6369	0.0	0.724	0.276	[best, gym, shoe, walk, stand, long, period, t...
696	Reebok Women's Fusion Flexweave Running Shoe, ...	Reviewed in the United States on 7 January 2019	DLR	Great Shoes for price	Shoes are nice and comfortable. I use them as ...	True	0.7269	0.0	0.567	0.433	[shoe, nice, comfort, use, walk, shoe]

The reviews can also be viewed in text format as shown below.

In [94]:
```python
# Assuming 'reviews' is your DataFrame with a '% negative_reviews' column
for product_name, group in reviews.groupby('product_name'):
    negative_reviews_text = ' '.join(group['review_text'])

    wordcloud = WordCloud(width=800, height=400, background_color='white').generate(negative_reviews_text)

negative_reviews_text
```

Out[94]:
```
"I rant and rave to all my friends about these shoes! They are by far the most comfortable shoes I have ever bought. I’m an
orthodontic technician so I’m constantly up and down on my feet for 10+ hours a day depending on the day. I have nev
er had a pair of shoes that helped my feet, legs, and back from hurting after a long shift till I bought these. Best part is
I bought a second pair for my everyday shoes also! “My New Lightweight Sneakers “ I a size 8 Medium, I received these today,
I tried the shoes on, the shoes have nice arch support, lightweight, comfortable, nice mauve color, I bought a size 8, my sh
oes fit true to size, I will purchase another pair in the future, the sneakers also make nice house slippers, also have mini
mal stretch, I received a nice compliment on my shoes.😊 I got these to wear around the house instead of going barefoot on
the hardwood. They are very comfortable and cute. For my purposes I had hoped they would slip on and off a bit easier. I sti
ll have to lean down to get the heel straightened out. I haven’t worn them out but I think they would be fine. I almost forg
ot I was wearing shoes because my feet get so much air when I wear these. It's been over 90° where I live and I wore these w
ithout any socks...when I took my shoes off my feet were COMPLETELY dry & not sweaty at all. They are so comfortable and fle
xible. Also standing for long periods have been easier. I just ordered 4 more pair.l, one for my mom, my daughter & two more
for yours truly. Very happy. These seemed to be made fairly decent, may even wear well, but while the fit was OK in length a
```

The words in the negative reviews can be viewed using the WordCloud library.

```
In [44]:  # WordCloud for words with negative reviews per product

          neg_tokens = [word for review in reviews_negative_subset['processed_review'] for word in review]

          wordcloud = WordCloud(background_color='white').generate_from_text(' '.join(neg_tokens))
          plt.figure(figsize=(10, 5))
          plt.imshow(wordcloud, interpolation='bilinear')
          plt.axis('off')
          plt.show()
```

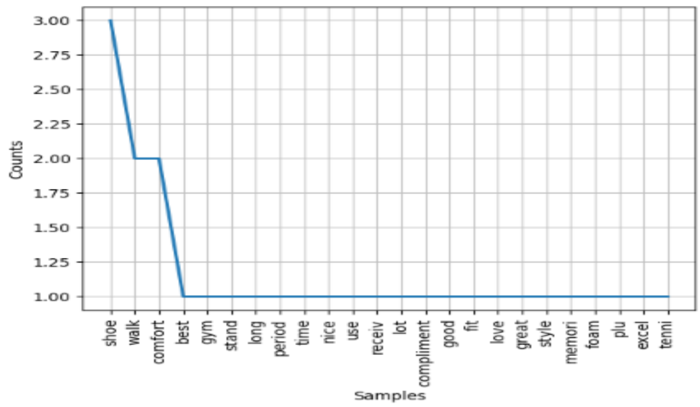
Using the WordCloud library for positive reviews

```
In [45]:  # WordCloud for words with positive reviews per product

          pos_tokens = [word for review in reviews_positive_subset['processed_review'] for word in review]

          wordcloud = WordCloud(background_color='white').generate_from_text(' '.join(pos_tokens))
          plt.figure(figsize=(10, 5))
          plt.imshow(wordcloud, interpolation='bilinear')
          plt.axis('off')
          plt.show()
```

We can also use the nltk function to understand the word frequencies as against just visualising it which can be difficult to interpret.

```
In [36]:  # Use the nltk FreqDist to underdtand word frequencies then tabulate
          from nltk.probability import FreqDist

          neg_freqdist = FreqDist(neg_tokens)

          neg_freqdist.tabulate(10)

   shoe    walk comfort    best     gym   stand    long  period    time    nice
      3       2       2       1       1       1       1       1       1       1
```

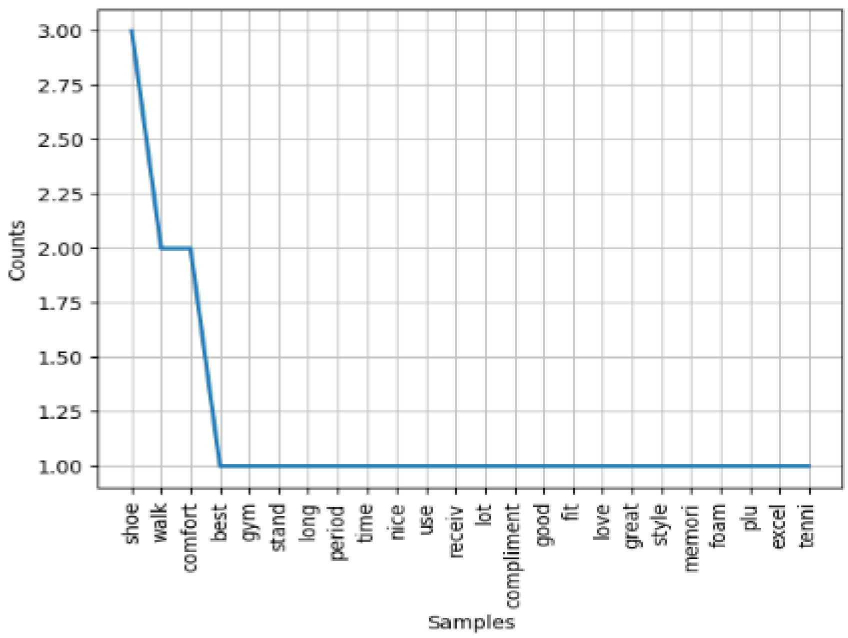
```
In [38]:  # Use the plot method to create a frequency distribution plot for the modt occuring words
          # in Zaptilla New Balance KV220 Navy/White, 9.5 UK Child and Ariat Mens Midtown 11 UK

          neg_freqdist.plot(30)
```



For positive reviews,

```
In [47]:  # Use the nltk FreqDist to underdtand word frequencies then tabulate
          from nltk.probability import FreqDist

          pos_freqdist = FreqDist(pos_tokens)

          pos_freqdist.tabulate(10)
```

```
   shoe    walk comfort    best     gym   stand    long  period    time    nice
      3       2       2       1       1       1       1       1       1       1
```




```
Out[48]: <Axes: xlabel='Samples', ylabel='Counts'>
```

Going by the word frequency uaing WordCloud, the visualised output is more difficult to interpret or understand, but with the nltk function, the word frequency is better understood.

Result Analysis and Discussion

In simple terms, text mining is used to structure unstructured data. Also known as Natural Language processing, it is used to extract meaningful information patterns and insights from unstructured datasets.

From the distribution of the reviews shown above, it is clear that most of the products have positive reviews compared with the negative and neutral ones. Some more insights that was gotten from the work done shows the percentage of the category of reviews.

References

1. Catherine C. (2021). *5 Principles of Data Ethics for Business.* https://online.hbs.edu/blog/post/data-ethics

2. Aditya K.P. (2020). *A Simple Explanation of K-Means Clustering.* https://www.analyticsvidhya.com/blog/2020/10/a-simple-explanation-of-k-means-clustering/