University of Salford, MSc Data Science

Module: Machine Learning & Data Mining

Session: Workshop Week 5

Topic: Classification (KNN and Decision Trees)

Tools: Jupyter Notebook

Objectives:

After completing this workshop, you will be able to:

- Explore, prepare, and pre-process a dataset for classification
- Use scikit-learn to implement the KNN algorithm for classification
- Use scikit-learn to implement the Decision Tree algorithm for classification Interpret common performance metrics including accuracy, precision, recall and F1 score.
- Evaluate and compare models' performance

MLDM Week 5 Workshop: Classification

Introduction:

Classification is a supervised learning approach in which a class label is predicted for a new observation based on training data. There are different algorithms for classification. In this workshop we will learn how to implement KNN and Decision tree.

An Example of a Business Problem:

One of the main applications of classification is marketing. Sometimes the product a company is advertising is not suitable for all people. For example, a person between the ages of 20 and 25 may like to spend more on smartphone covers than a person between the ages of 40 and 45. Likewise, a high-income person can afford to spend more on luxury goods than a low-income person. A business that wants to advertise a product likes to know which user will buy the product. This could be really helpful for the company selling their products. Predicting if a user is a potential customer or not is a classification problem.

Therefore, in this workshop we will apply classification algorithms to predict if a user can be a potential customer (will buy the advertised product) based on his/her attributes such as age, gender, etc.

Dataset:

The dataset we use in this workshop is "Social_Network_Ads.csv" that can be downloaded from Blackboard Week 5 folder. This dataset contains five columns:

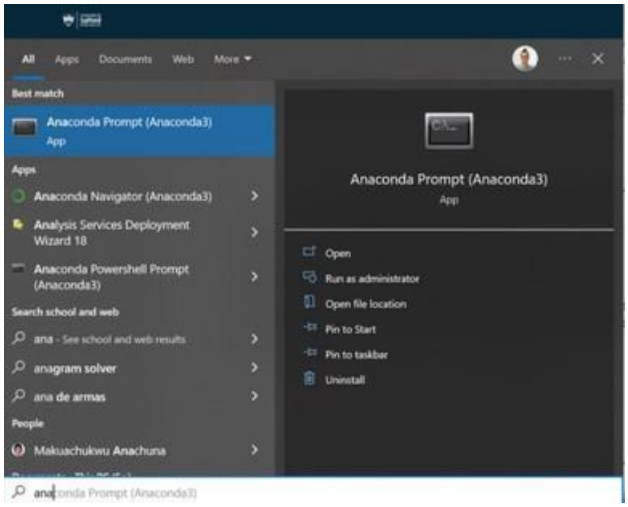
- User ID
- Gender
- Age
- Estimated Salary
- Purchased

The last column (i.e., Purchased) is the class label which gets two values. 1 shows the customer has purchased the product and 0 shows he hasn't bought the advertised product.

Important Note: Saving Your Notebook to F Drive

By default, when you launch Jupyter Notebook, the starting directory is in the C drive. If you are using a university desktop or laptop to complete this workshop, you should make sure that at the end of session you save your notebook to F drive so you can access it from other university devices.

Alternatively, you can launch Jupyter Notebook with the F drive as the starting directory. To do this, search for Anaconda Prompt in the Windows Search bar.



Once you have opened the Anaconda Prompt, enter the below command and press enter:

```
jupyter notebook --notebook-dir=F:
```

Part1: Creating the Notebook, Exploring and Pre-processing the Data

1) Download "Social_Network_Ads.csv" from Blackboard Week 5 folder.
2) Run Anaconda Navigator
3) Launch Jupyter notebook
4) In the Jupyter dashboard click on the 'New' button at the top right side to create a folder (Figure 1.). This folder will be named as Untitled Folder.

Figure 1. Creating a Folder

5) To rename the folder, click in the checkbox to left of the created folder and click on the Rename button in the top side of dashboard and enter Workshop5 as the new name.

Figure 2. Renaming the folder

6) In the Workshop5 folder click on the Upload button (Figure 3.) and upload "Social_Network_Ads.csv" you have saved before.

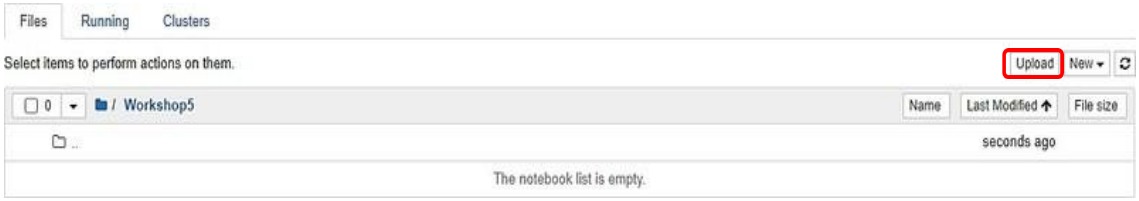


Figure 3. Uploading dataset from local computer to Jupyter

7) Click on the blue Upload button (Figure 4.). In this stage you have uploaded your dataset to Jupyter successfully.

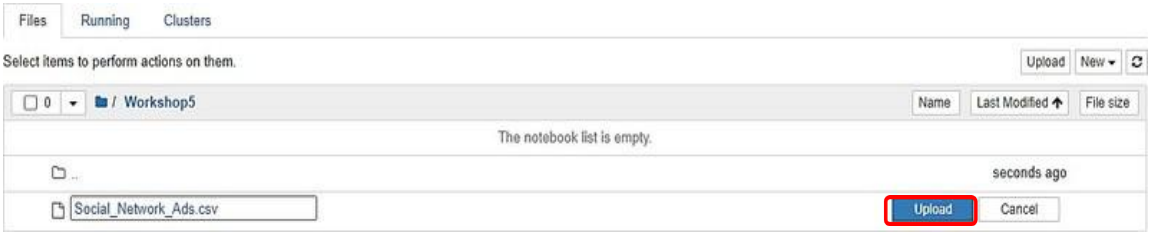


Figure 4. Completing dataset upload

8) Click on New button in Figure 4 and select Notebook to create a notebook in Workshop5 folder and rename it as KNN.

9) Import all required libraries using following codes:

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

10) Since your notebook and dataset are in the same folder, you can read the dataset using the following code. It reads the file into a pandas dataframe.

```
# Loading dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
```

11) Execute the following code. What does this statement do?

```
In [6]: dataset.shape
Out[6]: (400, 5)
```

12) We can explore the dataset using the following code:

(a) `dataset.head()`

It returns the header line and the first five rows of dataset.

Out[7]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

(b) The info() method prints a concise summary of the DataFrame. The information contains the number of rows, number of columns, column labels, column data types, the number of non-null values in each column, and memory usage.

```
In [9]: dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

(c) Execute the following code

```
In [10]: dataset.describe()
```

Out[10]:

	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

The describe() method returns a statistical description of the numerical columns in the DataFrame. It returns the following information for numeric columns:

- count - The number of not-empty values.
- mean - The average (mean) value.
- std - The standard deviation.
- min - the minimum value.
- 25% - The 25% percentile (The value at which 25% of the values are less than it.)
- 50% - The 50% percentile (The value at which 50% of the values are less than it.)
- 75% - The 75% percentile (The value at which 75% of the values are less than it.)
- max - the maximum value.

(d) If you want to return the above information for all columns of dataset, you should write the method as follows:

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

Out[12]:

	User ID	Gender	Age	EstimatedSalary	Purchased
count	4.000000e+02	400	400.000000	400.000000	400.000000
unique	NaN	2	NaN	NaN	NaN
top	NaN	Female	NaN	NaN	NaN
freq	NaN	204	NaN	NaN	NaN
mean	1.569154e+07	NaN	37.655000	69742.500000	0.357500
std	7.165832e+04	NaN	10.482877	34096.960282	0.479864
min	1.556669e+07	NaN	18.000000	15000.000000	0.000000
25%	1.562676e+07	NaN	29.750000	43000.000000	0.000000
50%	1.569434e+07	NaN	37.000000	70000.000000	0.000000
75%	1.575036e+07	NaN	46.000000	88000.000000	1.000000
max	1.581524e+07	NaN	60.000000	150000.000000	1.000000

Run the codes of part (a) to (d) in Jupyter. What kind of information do you get about "Social_Network_Ads.csv" dataset? How can this information help you in processing your data?

13) As we have explained in Workshop1, seaborn provides a variety of plot types useful for statistical data exploration. So, you can use seaborn library to show statistical information about different columns of dataset. Executing the below code, plot the distribution of column Age (Figure 5).

```
sns.histplot(dataset.Age)
plt.title('Age distribution')
plt.show()
```

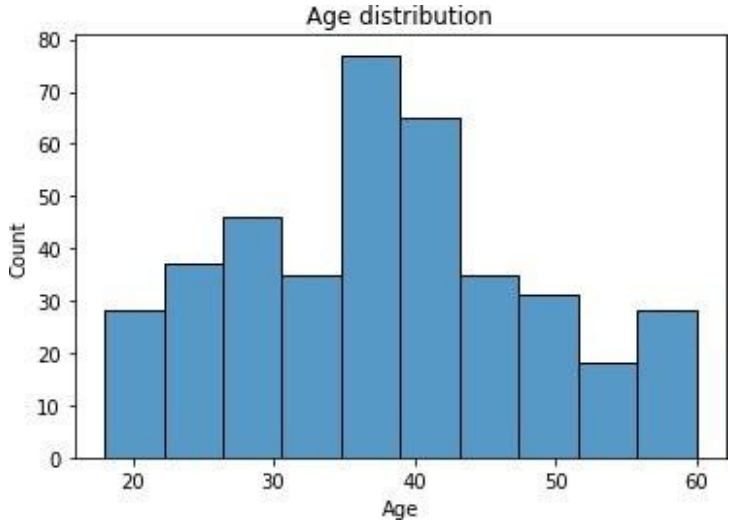


Figure 5. Age distribution in Social_Network_Ads dataset

14) We can explore some important patterns in the dataset. For example, by the code below we can explore the age distribution of the people who have responded to the social media ads and bought the product and those who haven't.

```
plt.figure(figsize=(10, 5))
plt.title("Product purchasing based on the age")
sns.histplot(x="Age", hue="Purchased", data=dataset)
plt.show()
```

If you execute the above code, you can see Figure 6. as the result.

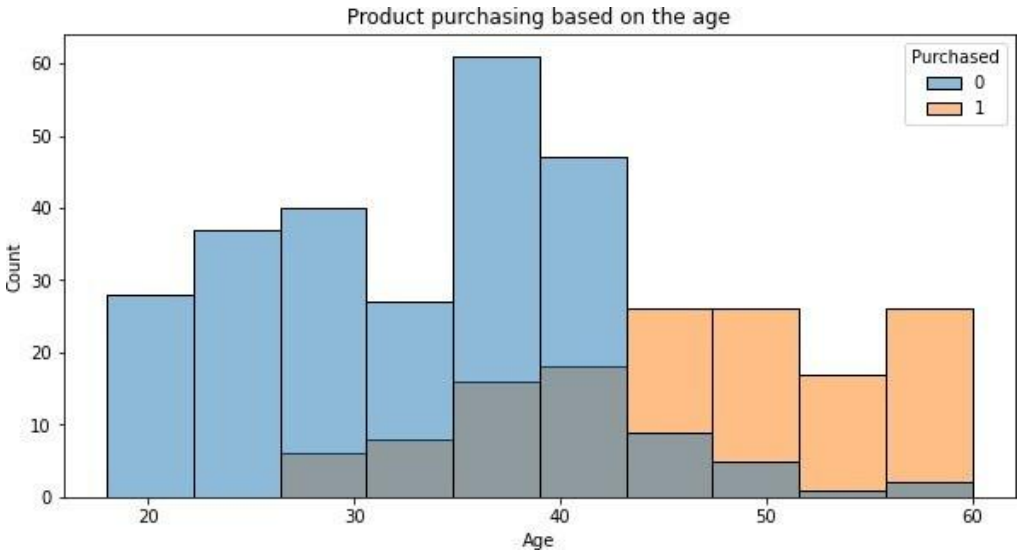


Figure 6. Product purchasing based on age

The visualization in Figure 6. shows that people over 40 among the target audience are more interested in purchasing the advertised product.

15) In our dataset, Gender has categorical value. We can use the code below to convert it to numerical value. It will replace Male by 0 and Female by 1 in the Gender column (You can use the methods you learned for categorization in Workshop2 as well)

```
dataset['Gender']=dataset['Gender'].replace(['Male','Female'],[0,1])
```

If you execute dataset.head(), you can see the following result:

	User ID	Gender	Age	Estimated Salary	Purchased
0	15624510	0	19	19000	0
1	15810944	0	35	20000	0
2	15668575	1	26	43000	0
3	15603246	1	27	57000	0
4	15804002	0	19	76000	0

Part 2: Classification

As we said earlier, social media ad scenario is a classification problem. The objective is to classify target audience and to predict if a user will buy the advertised item or not. In this section, we will employ scikit-learn library for classification. We will apply two different classification algorithms, i.e., KNN and Decision Tree on our dataset.

- **KNN:**

(a) Determining the class feature and input features:

In our problem the Purchased column is the class label or dependent variable, and other columns (except UserId) are independent variables or input features.

In the classification, we aim to predict the class label of a sample (output or y) based on its features (input or X). So, in the first step, we should slice our data into input and output. UserId has no effect on whether the user would purchase the item, so we don't include it in X.

Execute the following code to determine class feature and input feature:

```
X = dataset.iloc[:, [1, 2, 3]].values
y = dataset.iloc[:, 4].values
```

(b) Splitting the dataset into the Training set and the Test set:

We should split our data to train and test dataset. You can use the following code for data splitting:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
```

The test size can be float or int or it can be equal to None. If it is float, it should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split. If it is int, it will represent the absolute number of test samples. If None, the value is set to the complement of the train size. If train_size is also None, the test_size will be set to 0.25. The random state hyperparameter in the train_test_split() function controls the shuffling process. With random_state=None, we get different train and test sets across different executions. If a fixed value is assigned to it like random_state=0, we get the same train and test sets across different executions.

(c) Scaling features:

A dataset usually contains features of various dimensions and scales. Different scales of data features can affect the modeling of a dataset. Many machine learning algorithms such as KNN perform better when numerical input variables are scaled to a standard range. Standardization scales each input variable separately by subtracting the mean and dividing by the standard deviation to shift the distribution to have a mean of zero and a standard deviation of one.

We should scale the training data by subtracting the mean and dividing by the standard deviation. We should use the same mean and standard deviation for scaling test data as well.

fit_transform() method in Python is used on the training data so that we can scale the training data and also learn the scaling parameters of that data (i.e., the mean and variance of each of the features in the training data). Then, scaling should be applied to the test data too. The **transform()** method helps us in this case and scale test data by the same mean and variance that is calculated from our training data.

The Following code standardizes the train and test dataset.

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

(d) Training the model:

It is not necessary to implement KNN code from scratch. You can use KneighborsClassifier from scikit learn library.

```
# Fitting K-NN to the Training set
from sklearn.neighbors import KNeighborsClassifier
classifier=KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2)
classifier.fit(X_train_s, y_train)
```

p in the above code shows the power parameter for the Minkowski metric. When p = 2, this is equivalent to euclidean_distance.

(e) Evaluating the model:

Once the model is trained, we can use the 'predict' function on our model to make predictions on our test data.

```
# Predicting the Test set results
y_pred=classifier.predict(X_test_s)
print(y_pred)
```

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

y_pred shows the predicted class (label) for test dataset. Print the real value of labels in test dataset and compare them with predicted values:

```
print(y_test)
```

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

To evaluate the performance of model, you can use the following code:

```
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:0.92


Confusion Matrix:
[[73  6]
 [ 4 37]]


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

              precision    recall  f1-score   support

           0       0.95      0.92      0.94        79
           1       0.86      0.90      0.88        41

    accuracy                           0.92       120
   macro avg       0.90      0.91      0.91       120
weighted avg       0.92      0.92      0.92       120
```

The confusion_matrix function generates a confusion matrix using the generated predictions (y_pred) and the true class labels for the test data (y_test). It shows the number of TP, FN, FP and TN.

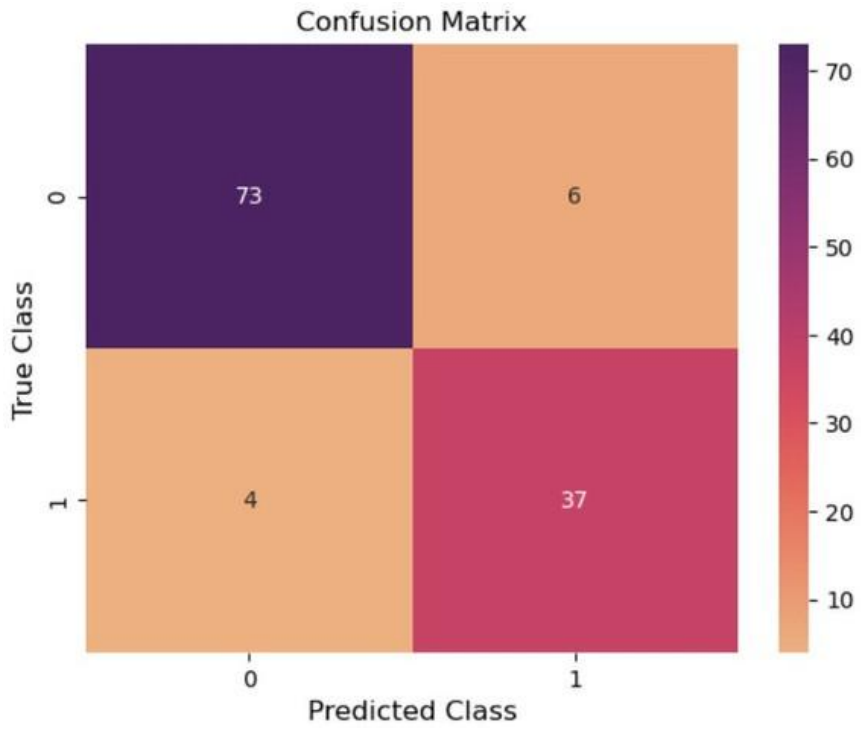
The classification_report function shows different metrics such as precision, recall, f1-score, etc., for model evaluation

Also, we can use the seaborn heatmap to visualise the confusion matrix.

```
ax = sns.heatmap(cm, cmap='flare',annot=True, fmt='d')

plt.xlabel("Predicted Class",fontsize=12)
plt.ylabel("True Class",fontsize=12)
plt.title("Confusion Matrix",fontsize=12)

plt.show()
```



- **Decision Tree:**

  All parts are similar to KNN except KNN section- part(d). In fact, it is enough to change our model from KNN to decision tree. In addition for decision tree, remove scaling (part c).

  To a new notebook, from the file menu, choose "Save Notebook As" (Figure 7.).

  Rename the notebook to "Decision-tree" and then cut/paste the created notebook in your folder.

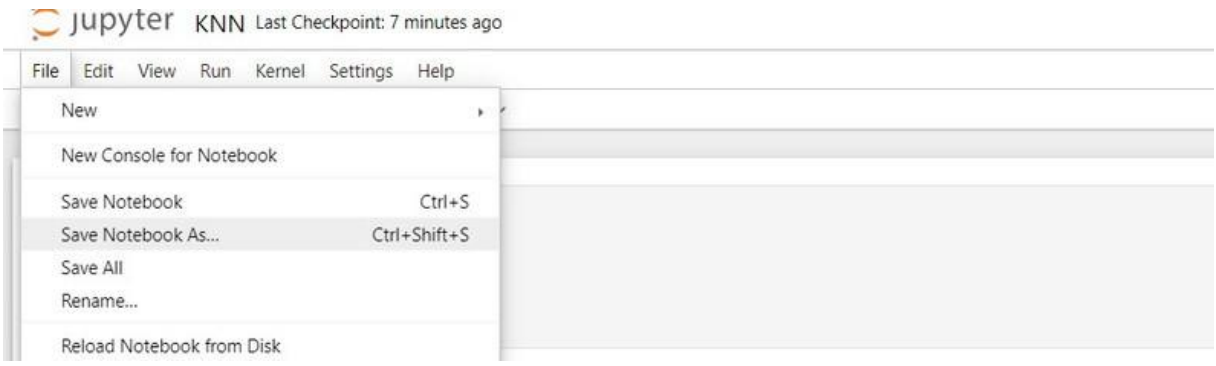


Figure 7. Make a copy of your notebook

Everything remains the same. You only should remove part c (scaling) and replace the code in KDD(d) section (Training the model) by the below code:

```
# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

From the Kernel menu, select “Restart & Run All cells” and see the result for Decision tree classifier.

Are the results different from KNN? Is there a significant difference?