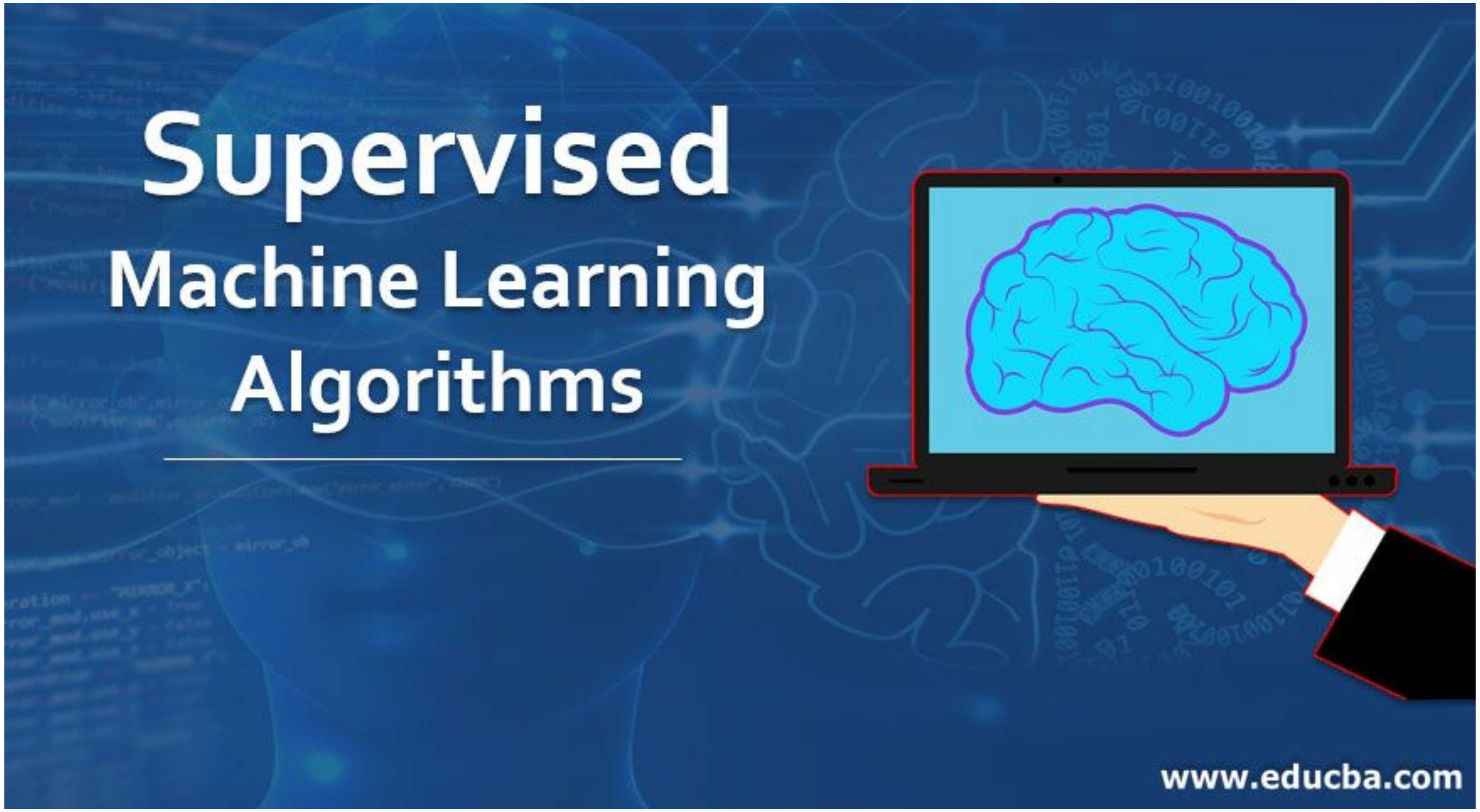


ECOM SCHOOL

המכללה למקצועות הדיגיטל וההייטק

Last lecture reminder

We learned about:

- Logistic Regression - Python example
- Classification Metrics - Introduction
- Classification Metrics - The confusion matrix
- Classification Metrics - Accuracy and Accuracy Paradox
- Classification Metrics - Precision, Recall, F1-Score
- Logistic Regression with Classification Metrics - Python Example

ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

Logistic Regression - Multiple Categories

Until now we only saw how Logistic Regression is dealing with binary category dataset (when we have only 2 categories).
However, Logistic Regression has the ability to also provide prediction on categorical datasets with multiple categories.

One-vs-All technique → In case of multiple categories logistic regression using the One-vs-All technique meaning each iteration a different category will be evaluate as ‘success’ and the other categories as ‘failure’. So for `k` categories of the label, `k` binary logistic models are built where each model classifies instances of one category as positive and all other categories as negative. Then the model, that fetches the highest probability, classifies the multi-class instance according to that class.

Note: When dealing with One-vs-All technique it’s important to use penalty like L2 in order to prevent overfitting.

Logistic Regression - Multiple Categories

For example → Let's say, you've 3 classes - A, B and C, and you following model using the One-vs-All approach, We know that with 3 classes we will generate 3 different binary models:

- **Model1** → Trained to recognize Class A vs Classes B and C
- **Model2** → Trained to recognize Class B vs Classes A and C
- **Model3** → Trained to recognize Class C vs Classes A and B

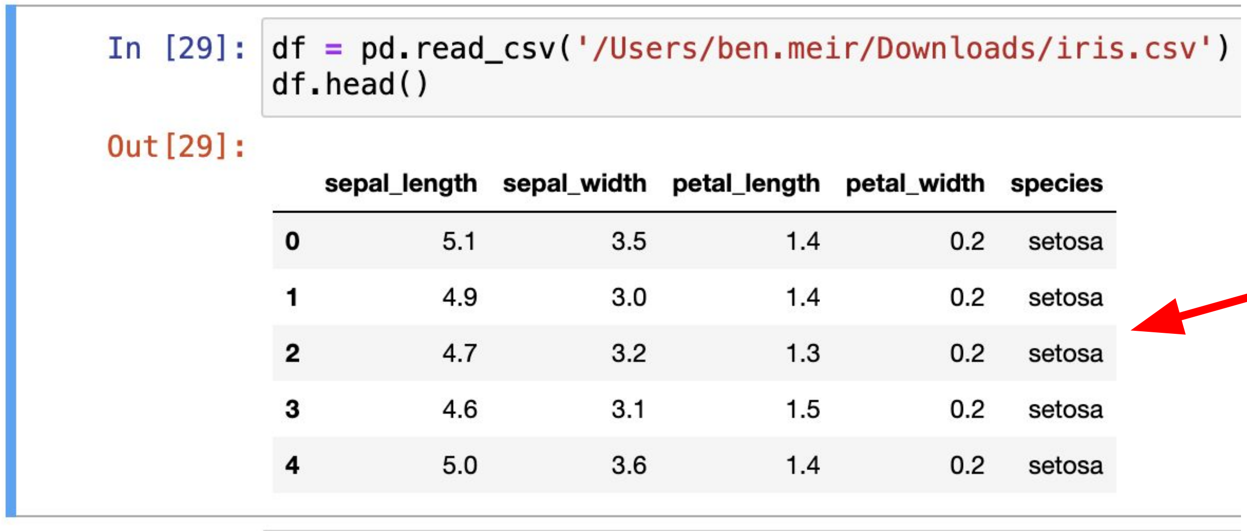
Now when you want to classify a new instance, You'd input the instance into each of these models.
Each model would output a probability that the instance belongs to its corresponding class.
You'd then take the maximum of these probabilities and classify the instance to the class with the highest probability.

So, if Model1 outputs a probability 0.2, Model2 outputs a probability of 0.7, and Model3 outputs a probability of 0.4, then the instance would be classified as class B (since the Model2, which identifies B vs rest, outputs the highest probability).

Logistic Regression Multiple Categories - Python Example

In order to explore this ability we will use the **‘iris.csv’** file continuing the famous iris dataset.

The iris dataset → This is one of the most famous and widely used datasets in machine learning. The iris dataset contains measurements for 150 iris flowers from three different species. The species are Iris Setosa, Iris Versicolour, and Iris Virginica.
So the iris dataset provide 4 different features for each category and has 3 different categories in total which is the definition of multi-categorical dataset.



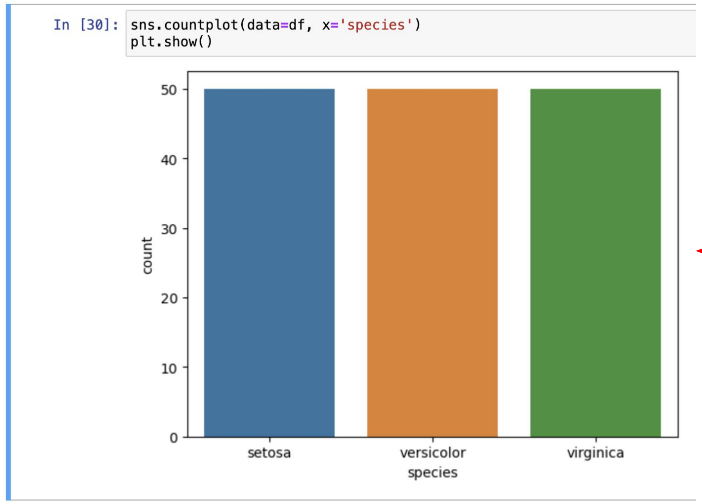
```
In [29]: df = pd.read_csv('/Users/ben.meir/Downloads/iris.csv')
         df.head()
Out[29]:
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

The ‘iris.csv’ dataset provide data about the length and width of the petal (the inner part of the flower) and the length and width of the sepal (the outer part of the flower).
For each data row the dataset provide the iris category it belongs to.

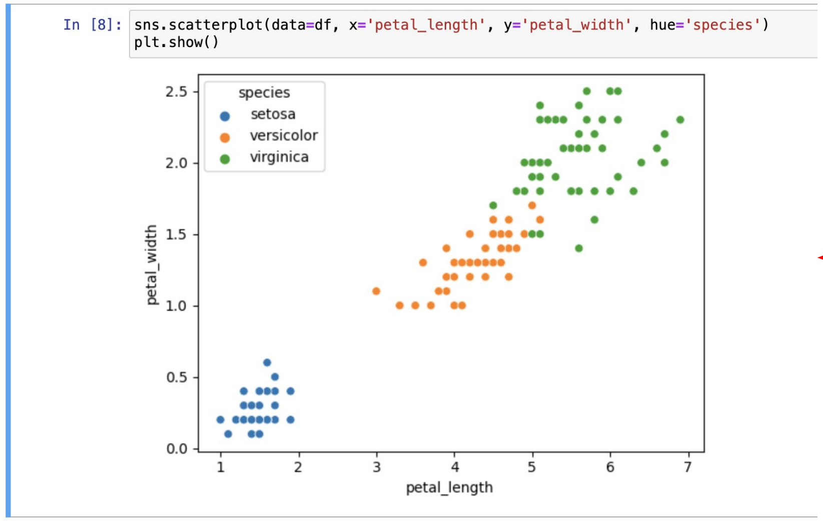
Logistic Regression Multiple Categories - Python Example

We can also explore our ‘iris’ dataset using seaborn plots:



```
In [30]: sns.countplot(data=df, x='species')
         plt.show()
```

We can see that we have 3 different species categories and in each of them we have 50 observations.
The total observations in the dataset is 150



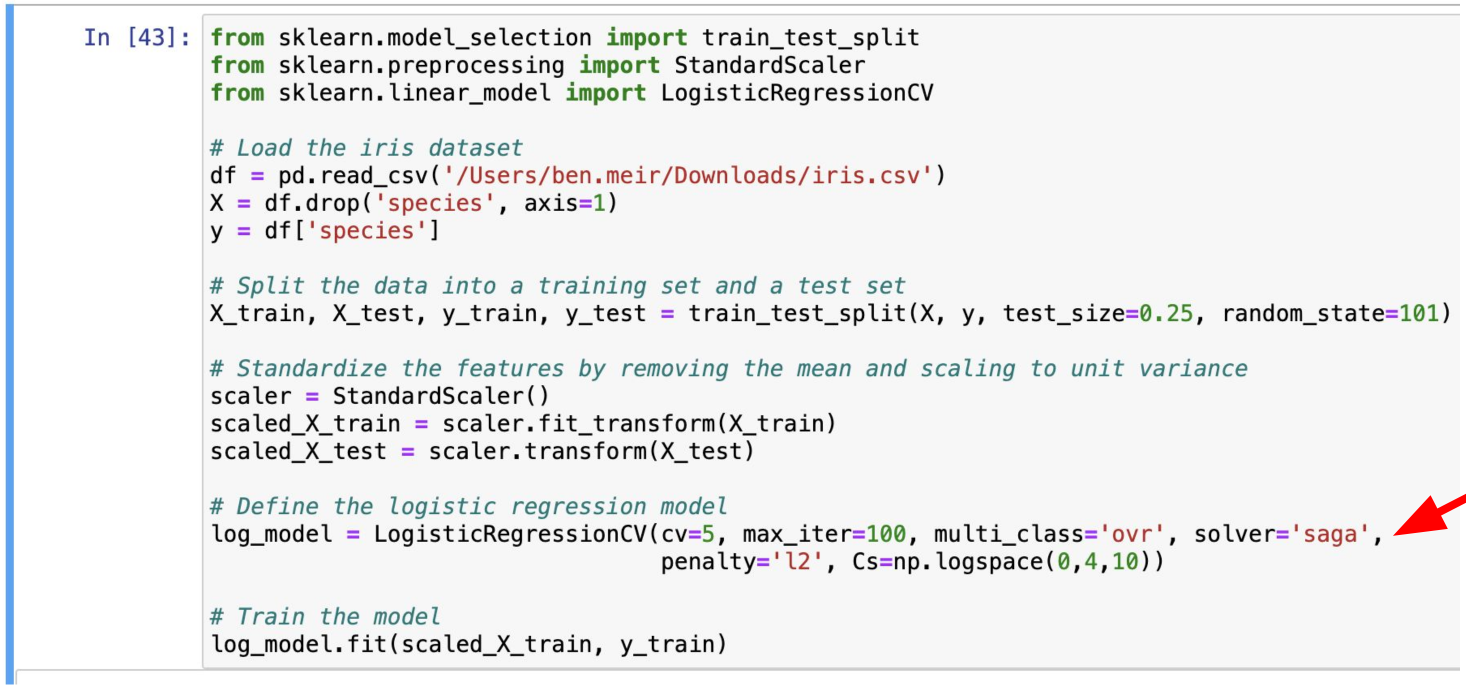
```
In [8]: sns.scatterplot(data=df, x='petal_length', y='petal_width', hue='species')
        plt.show()
```

We can see that the ‘setosa’ spicy is very different from the other two regarding its features so it should be more easy to our model to successfully predict this category.

Logistic Regression Multiple Categories - Python Example

Now let's use Logistic Regression model on the multiple categories 'iris' dataset:

For this model we will use the **LogisticRegressionCV** model because our model is now more complex (more categories to examine) so we want to make sure we protect our model from unwanted bias and imbalance categories.



```
In [43]: from sklearn.model_selection import train_test_split
         from sklearn.preprocessing import StandardScaler
         from sklearn.linear_model import LogisticRegressionCV

         # Load the iris dataset
         df = pd.read_csv('/Users/ben.meir/Downloads/iris.csv')
         X = df.drop('species', axis=1)
         y = df['species']

         # Split the data into a training set and a test set
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=101)

         # Standardize the features by removing the mean and scaling to unit variance
         scaler = StandardScaler()
         scaled_X_train = scaler.fit_transform(X_train)
         scaled_X_test = scaler.transform(X_test)

         # Define the logistic regression model
         log_model = LogisticRegressionCV(cv=5, max_iter=100, multi_class='ovr', solver='saga',
                                          penalty='l2', Cs=np.logspace(0,4,10))

         # Train the model
         log_model.fit(scaled_X_train, y_train)
```

We are using LogisticRegressionCV model with the provided parameters (explanation of the parameters will be in the next slide)

ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

Logistic Regression Multiple Categories - Python Example

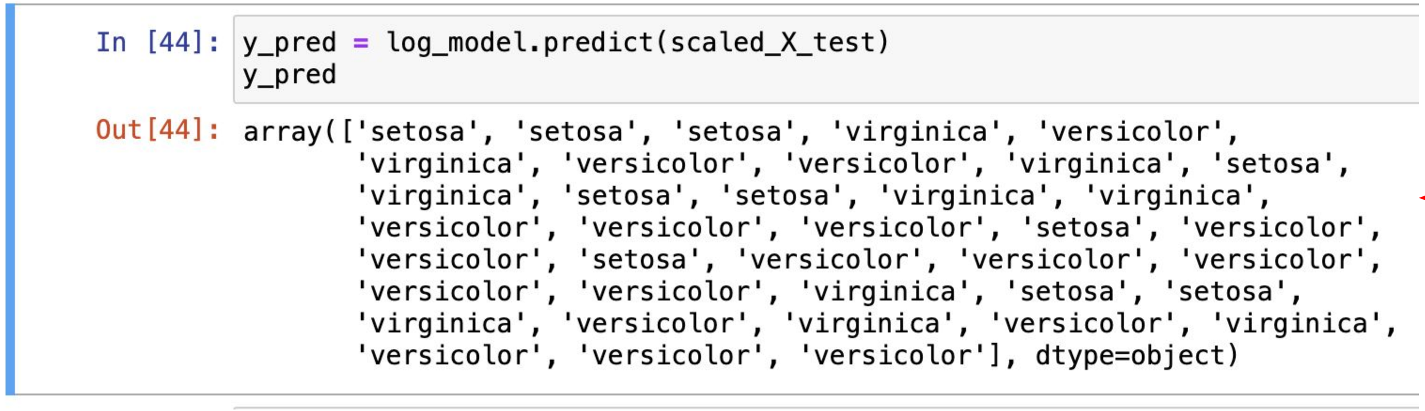
Let’s explain our LogisticRegressionCV parameters:

- **‘cv’** → The number of folds in the cross-validation. In this case, using a value of 5 means the training set will be split into 5 separate subsets and the model will be trained and tested 5 times, each time on a different subset.
- **‘max_iter’** → The maximum number of iterations for the solver to converge to the solution.
- **‘multi_class’** → The technique we want to handle multi-class problems. 'ovr' stands for one-vs-rest (one-vs-all) which means a binary problem is fit for each label.
- **‘solver’** → The algorithm to use for the optimization problem, in one-vs-all technique with regularization its best practice to use the ‘saga’ solver algorithm.
- **‘penalty’** → The type of regularization applied to the model. 'l2' means it uses L2 regularization.
- **‘Cs’** → The array of optional C (lambda) values for the L2 penalty.

ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

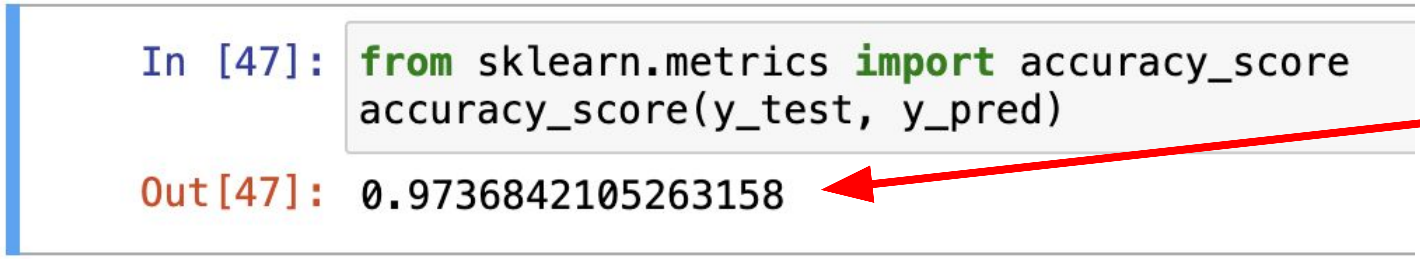
Logistic Regression Multiple Categories - Python Example

Once we finish train our model we can examine the model prediction results using the classification metrics we learned:



```
In [44]: y_pred = log_model.predict(scaled_X_test)
         y_pred

Out[44]: array(['setosa', 'setosa', 'setosa', 'virginica', 'versicolor',
                'virginica', 'versicolor', 'versicolor', 'virginica', 'setosa',
                'virginica', 'setosa', 'setosa', 'virginica', 'virginica',
                'versicolor', 'versicolor', 'versicolor', 'setosa', 'versicolor',
                'versicolor', 'setosa', 'versicolor', 'versicolor', 'versicolor',
                'versicolor', 'versicolor', 'virginica', 'setosa', 'setosa',
                'virginica', 'versicolor', 'virginica', 'versicolor', 'virginica',
                'versicolor', 'versicolor', 'versicolor'], dtype=object)
```

We can see that our model predict the string category for each row in the test dataset. We don't need to convert the label to numeric number, the model can handle it as a string.



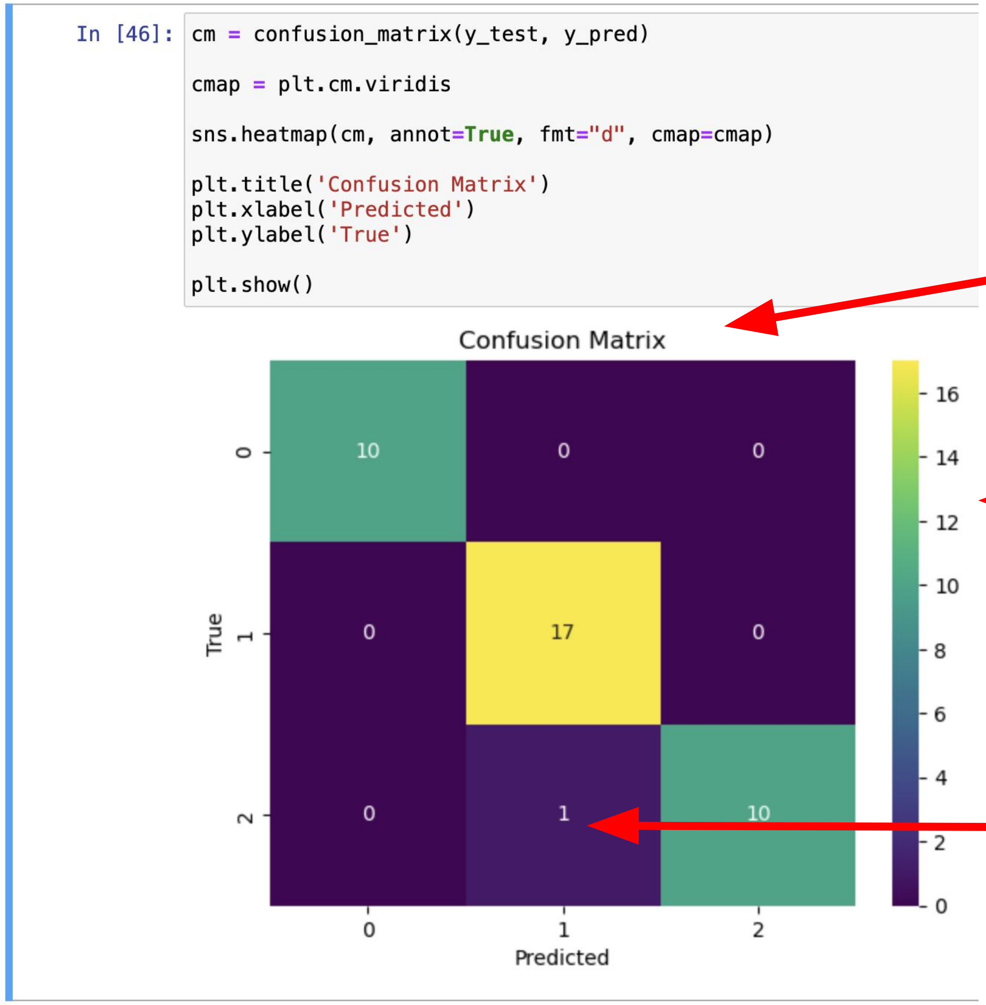
```
In [47]: from sklearn.metrics import accuracy_score
         accuracy_score(y_test, y_pred)

Out[47]: 0.9736842105263158
```

Our model accuracy score is 0.973 meaning it predicted successfully 97.3% of the observations

ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

Logistic Regression Multiple Categories - Python Example

And finally, we can also generate the confusion matrix to see our model results:



```
In [46]: cm = confusion_matrix(y_test, y_pred)

         cmap = plt.cm.viridis

         sns.heatmap(cm, annot=True, fmt="d", cmap=cmap)

         plt.title('Confusion Matrix')
         plt.xlabel('Predicted')
         plt.ylabel('True')

         plt.show()
```

	0	1	2
0	10	0	0
1	0	17	0
2	0	1	10

The class order numbers is:
- Setosa: 0
- Versicolor: 1
- Virginica: 2

Our confusion matrix now has 3 rows and 3 columns because we have 3 different categories.
The diagonal of the matrix is where the model predict correctly.

According to the confusion matrix, our model missed only 1 prediction from the entire test dataset. (Our model predict 'Versicolor' class while the real class was 'Virginica')

Class Exercise - Logistic Regression

Instructions:

Use the **‘penguins’** dataset provided by Seaborn library, Your mission to predict the species of penguins based on selected features.

perform Multi-class Logistic Regression machine learning modeling with the following instructions:

- Get the ‘penguins’ data set by running the following command:
  penguins = sns.load_dataset('penguins')
- Create a new dataset continuing only the following columns: **‘sex’, ‘body_mass_g’, ‘island’, ‘species’**
  The new dataset will provide data about each penguin. Each row will provide date about the penguin sex, it’s body mass in grams and the island it leave at. In addition each row will provide data about the penguin type (‘Adelie’, ‘Chinstrap’, ‘Gentoo’).
- Apply data processing and remove rows with ‘null’ values in any column

Class Exercise - Logistic Regression

Instructions:

- Change the ‘sex’ and ‘island’ string values to numeric values so the machine learning model will be able to use the data. For example ‘male’ will be 0 and ‘female’ will be 1. (hint → use pd.get_dummies())
- Apply standardization feature scaling
- Apply cross-validation Logistic Regression to predict the type of new penguin according to the provided feature values.
- Use the ‘One-vs-All’ technique the ‘saga’ solver algorithm and L1 penalty
- Print your model prediction, Accuracy, Precision, Recall and F1-score metrics
- Plot your model confusion matrix
- In case your model is not accurate enough, think about ways to improve it (no need to change the code but just think about actions that could improve your model prediction)

Class Exercise Solution - Logistic Regression

ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

KNN - Introduction

KNN → KNN stands for **K-Nearest Neighbors.** It is one of the simplest machine learning algorithms typically used for classification and regression. The algorithm works on the principle of identifying the 'k' number of points in the dataset that are closest to the point which we want to classify or predict, hence the name 'K-Nearest Neighbors'.

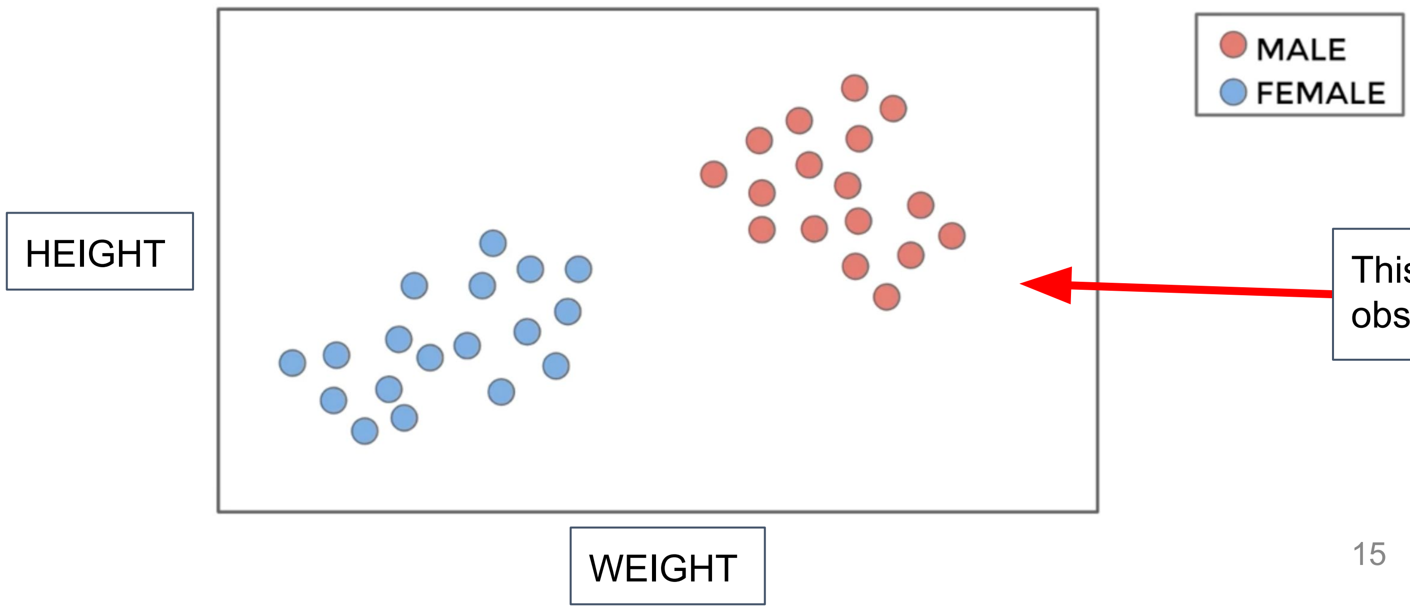
The KNN algorithm is under the category called **"instance-based learning"**, or **"lazy learning"**, because it does not learn a model from the training data but memorizes the training dataset instead.

Note: KNN is a distance algorithm and because it relaying on the distance between the observations in order to predict the next observation, it is very important to perform feature scaling before using this algorithm.

KNN - Theory

To better understand how KNN algorithm is working let’s take a simple example:

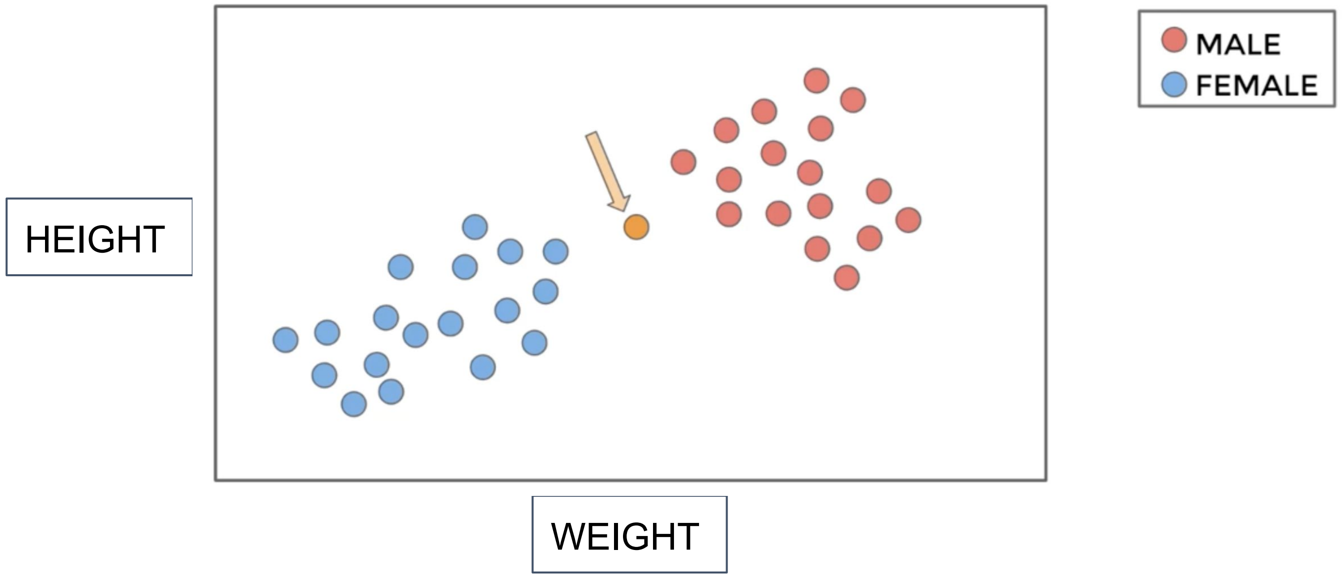
Let’s say we have dataset about the height, weight and sex of a specific baby animal and we want to write a machine learning model that will predict the animal sex according to those features.
This task is a binary classification task because we have only 2 categories to predict (male or female).
We want to use the KNN algorithm in order to predict the animal sex.



This is a scatter plot with our current observations

KNN - Theory

Now let’s say we want to predict new observation using KNN:

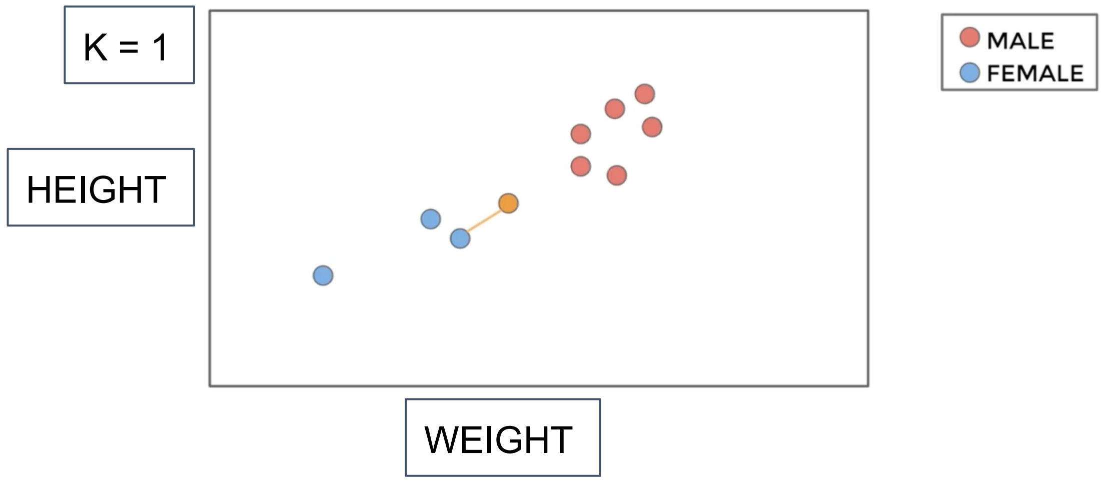


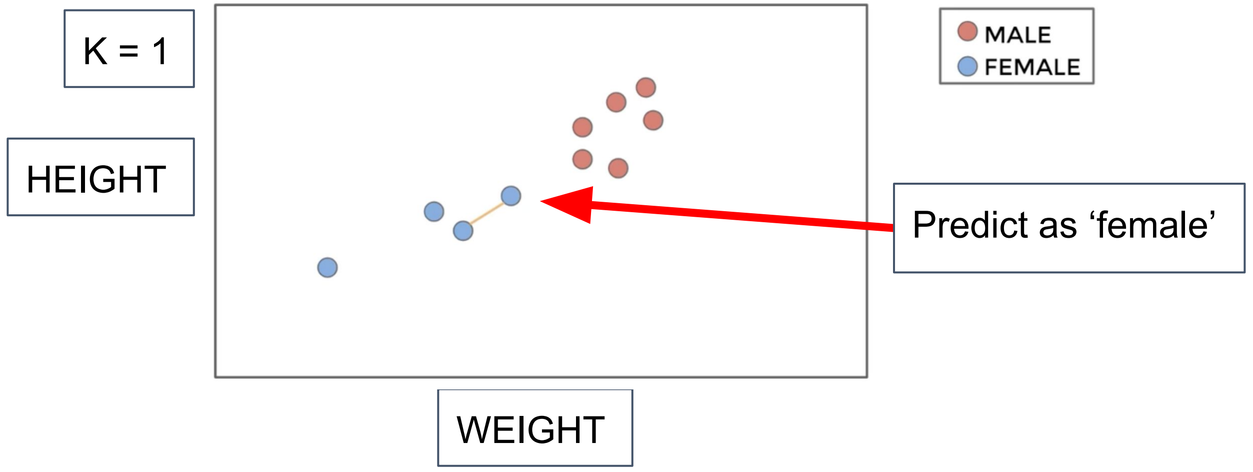
In KNN algorithm, the prediction for the unknown observation will be determined by the **K-Nearest Neighbors**, so first we need to determine what will be the K value.

KNN - Theory

Let's remove some observations from the scatter plot so it will be better to understand the algorithm.

For **K = 1** (meaning we are looking only at 1 nearest neighbor and the prediction will be according to the neighbor).



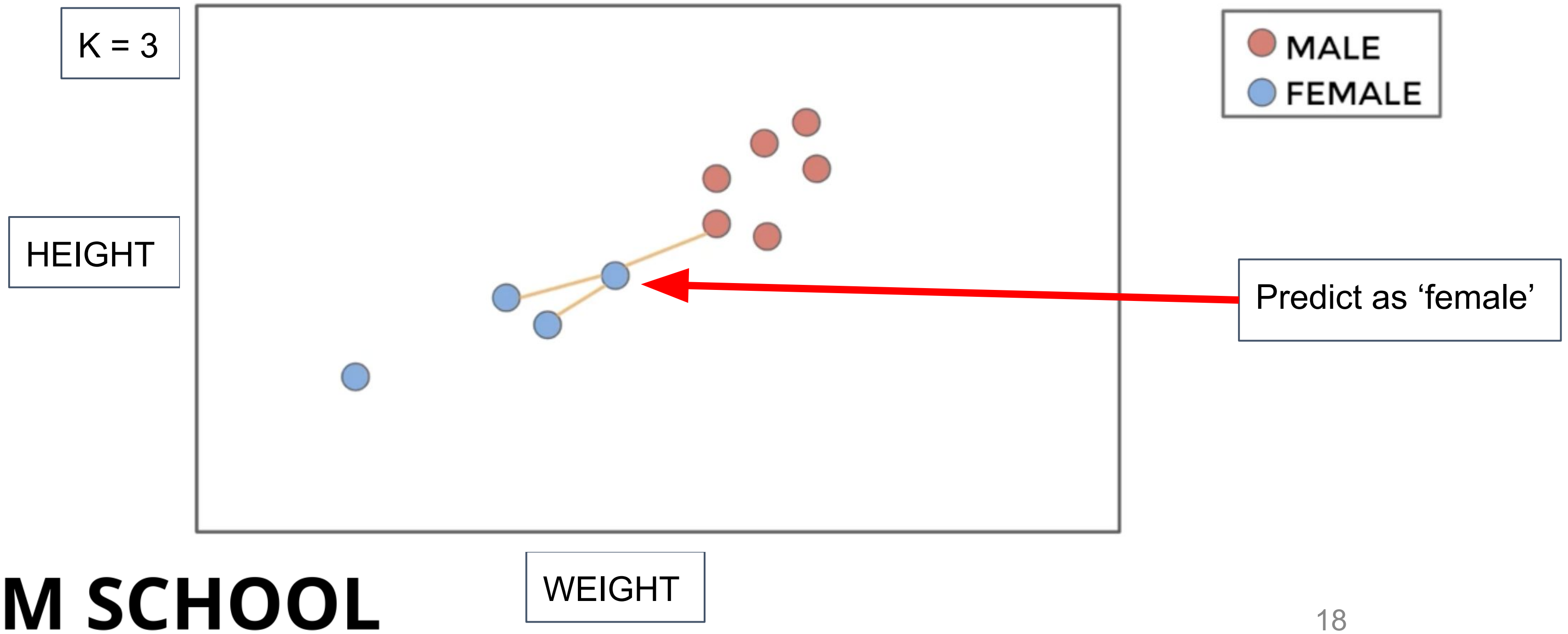


We can see that the algorithm predict the unknown observation as 'female' because we are using K = 1 and the first nearest neighbor is sex is female.

KNN - Theory

The same way we can increase the K value so the prediction will rely on more neighbors. The prediction will be determine according to the higher neighbor count.

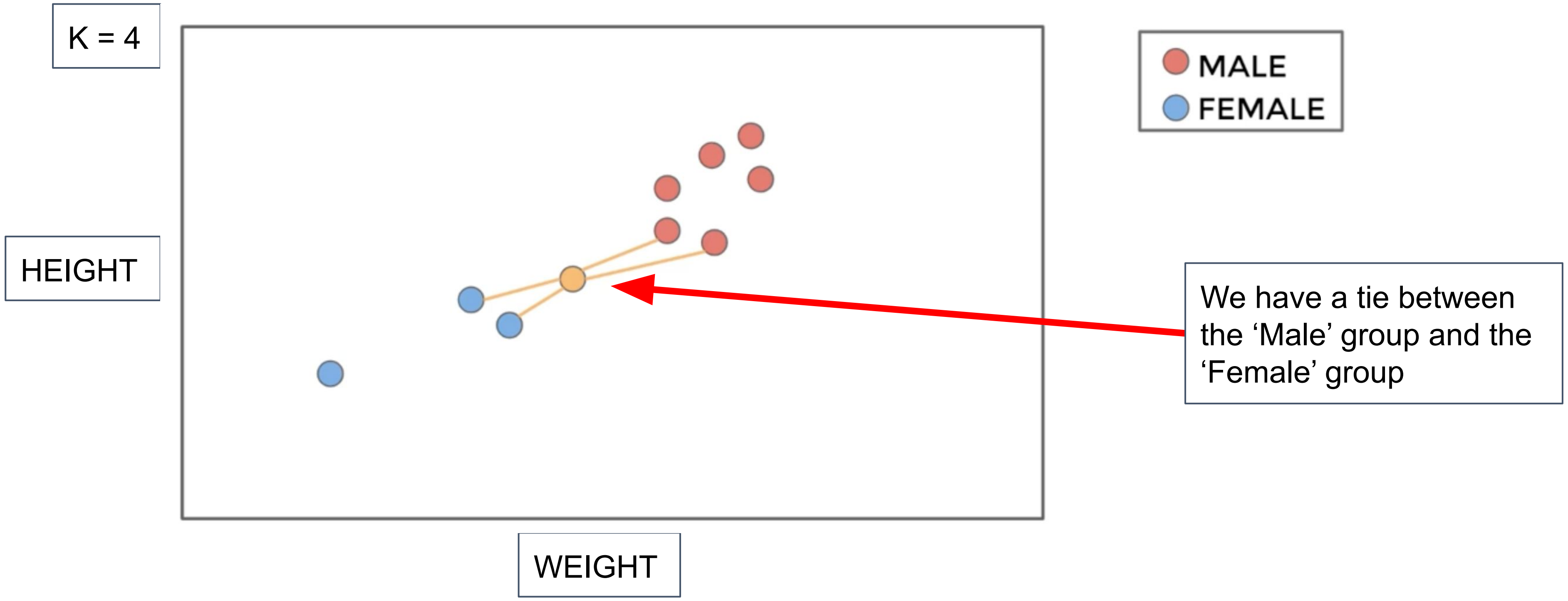
For example → If we will take **K = 3** we will have 2 neighbors with ‘female’ value and 1 neighbor with ‘male’ value so the algorithm will predict ‘female’.



KNN - Theory

Because KNN takes the larger group of neighbors and predict according to this group, when dealing with even number of K it is possible to get a tie between the 2 groups.

For example → If we will take **K = 4** we will get a tie between the 'Male' group and the 'Female' group.


K = 4
HEIGHT
WEIGHT
● MALE
● FEMALE
We have a tie between the 'Male' group and the 'Female' group

ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

KNN - Theory

How we can deal with a tie in KNN algorithm?

- Always chose an odd K, this way it will not be possible to get the same number of nearest neighbors in both groups
- In case of a tie reduce K by 1 until the tie is broken
- Randomly choose a group to predict so the tie will break
- Choose the nearest class point as the prediction value (regardless the other k nearest neighbors)

Note: Scikit-Learn deal with tie between groups by selecting the most nearest neighbor and predict accordingly.

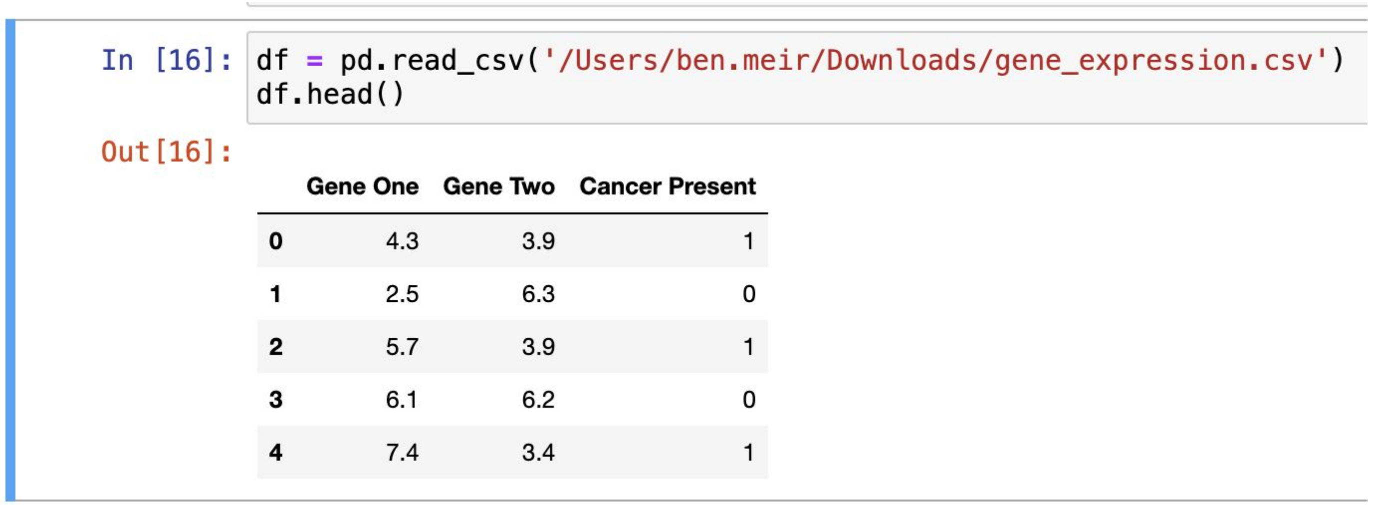
KNN - Python Example

For this KNN example we will use the **‘gene_expression.csv’** file.

This file provide data about patients gene ratio of 2 specific genes **‘Gene One’** and **‘Gene Two’**.

In addition the dataset provide data about cancer present in those patients when **1 = cancer is present** and **0 = cancer is not present**.

Our mission is to predict if cancer present according to ‘Gene One’ and ‘Gene Two’ ratio values in new patients. <u>For the prediction process we want to use the KNN supervised learning algorithm</u>.



```
In [16]: df = pd.read_csv('/Users/ben.meir/Downloads/gene_expression.csv')
         df.head()
Out[16]:
```

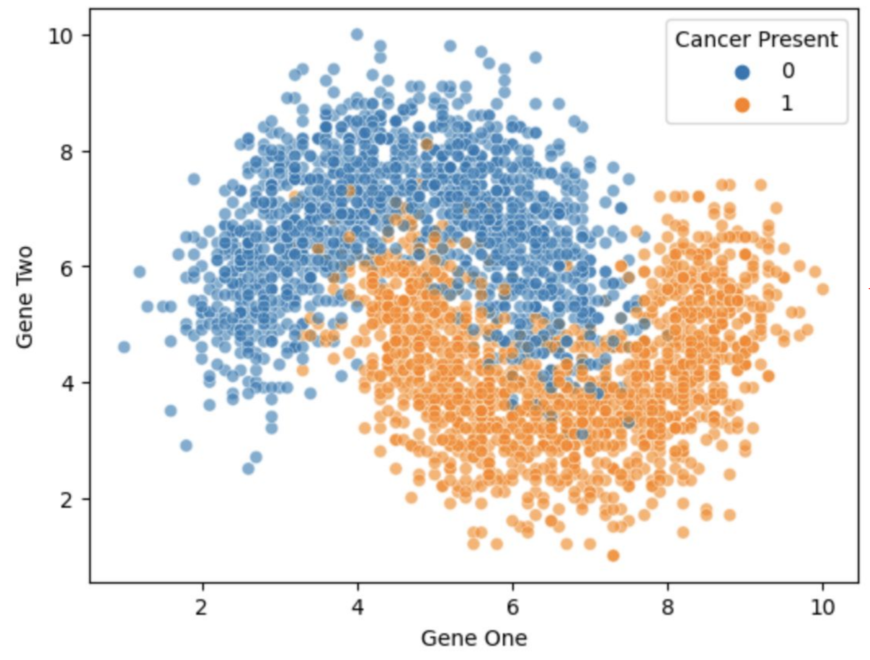
	Gene One	Gene Two	Cancer Present
0	4.3	3.9	1
1	2.5	6.3	0
2	5.7	3.9	1
3	6.1	6.2	0
4	7.4	3.4	1

KNN - Python Example

First let’s explore our dataset and see if we can see a correlation between higher value of ‘Gene One’ and ‘Gene Two’ and the present of cancer in patients.

For that we will generate a Seaborn scatterplot chart:

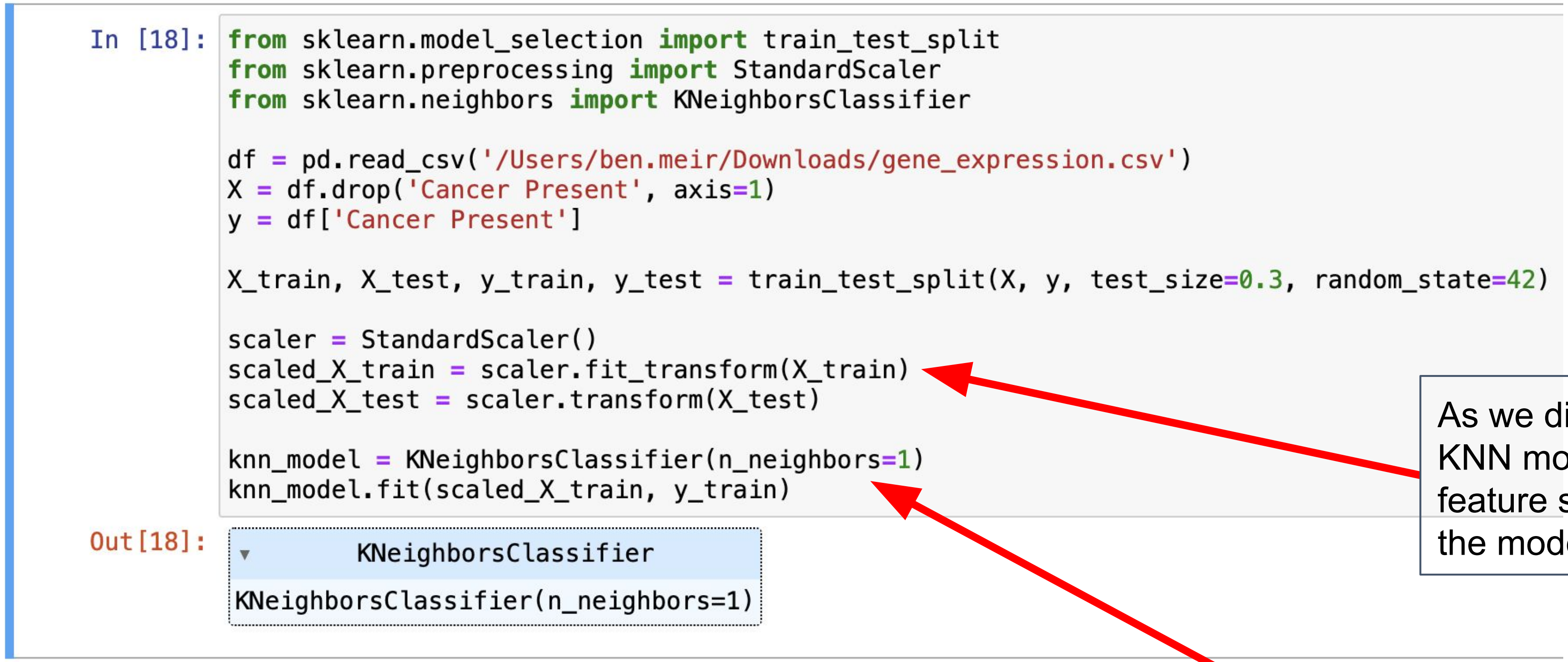
```
In [17]: sns.scatterplot(data=df, x='Gene One', y='Gene Two', hue='Cancer Present', alpha=0.6)
         plt.show()
```



We can see that patients with higher values of ‘Gene One’ and lower values of ‘Gene Two’ are more likely to have cancer

KNN - Python Example

Now, let's generate our simple KNN model with **K = 1** and evaluate our model accuracy using the categorial metrics we learned before.



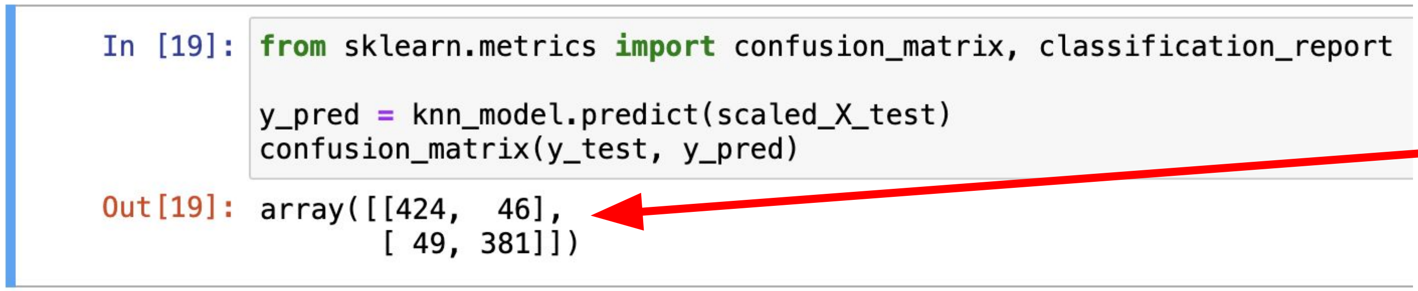
```
In [18]: from sklearn.model_selection import train_test_split
         from sklearn.preprocessing import StandardScaler
         from sklearn.neighbors import KNeighborsClassifier

         df = pd.read_csv('/Users/ben.meir/Downloads/gene_expression.csv')
         X = df.drop('Cancer Present', axis=1)
         y = df['Cancer Present']

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

         scaler = StandardScaler()
         scaled_X_train = scaler.fit_transform(X_train)
         scaled_X_test = scaler.transform(X_test)

         knn_model = KNeighborsClassifier(n_neighbors=1)
         knn_model.fit(scaled_X_train, y_train)

Out[18]: ▾ KNeighborsClassifier
         KNeighborsClassifier(n_neighbors=1)
```

As we discussed, when executing KNN model we always want to perform feature scaling before we start to train the model

We import the KNN algorithm from Scikit-Learn and select the n_neighbors (which is the K value) to be 1

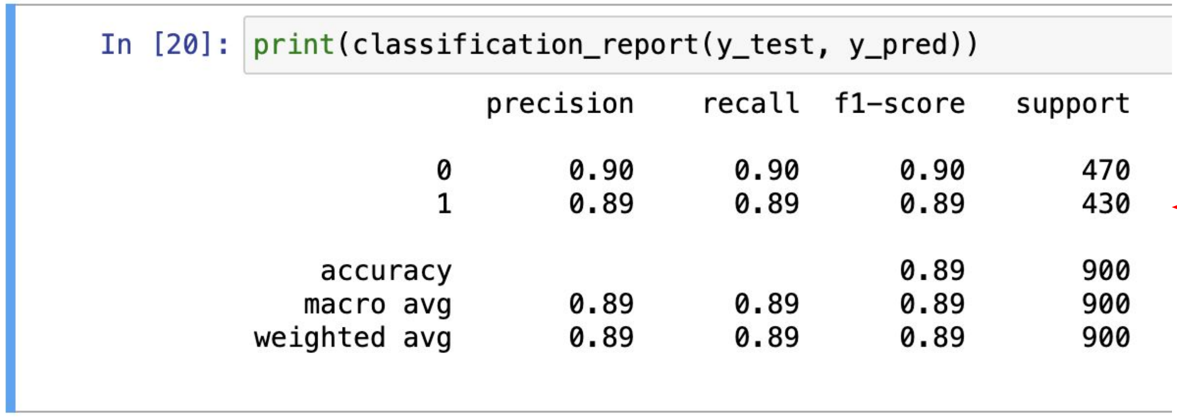
KNN - Python Example

Finally, let’s predict our testset data and use the classification metrics to evaluate our model accuracy.



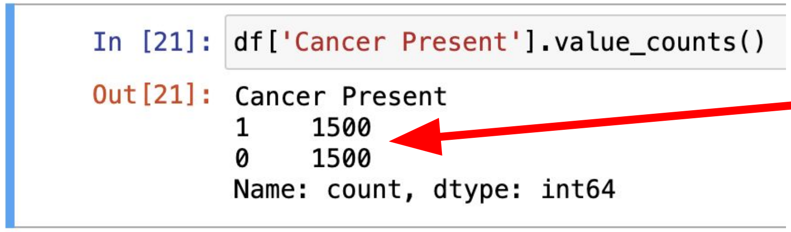
```
In [19]: from sklearn.metrics import confusion_matrix, classification_report

         y_pred = knn_model.predict(scaled_X_test)
         confusion_matrix(y_test, y_pred)

Out[19]: array([[424,  46],
                [ 49, 381]])
```

Our model predicted wrong predictions to 95 observation out of 900 observations in total



```
In [20]: print(classification_report(y_test, y_pred))

                       precision    recall  f1-score   support

                   0       0.90      0.90      0.90       470
                   1       0.89      0.89      0.89       430

            accuracy                           0.89       900
           macro avg       0.89      0.89      0.89       900
        weighted avg       0.89      0.89      0.89       900
```

We can see that our model perform pretty good regarding the precision, In addition we don’t see any problem with imbalanced classes



```
In [21]: df['Cancer Present'].value_counts()

Out[21]: Cancer Present
         1    1500
         0    1500
         Name: count, dtype: int64
```

We can also see that the number of observations belong to each group is completely equal, so there are no imbalance classes in our dataset