

## Predicting product sales through ads delivered on Social Networking Sites using K.N.N

## Computer Structure (INS 611)

**Final Presentation**

**MASTER OF INTERNET SYSTEMS**

**SUBMITTED TO**

**Dr. Innocent KABANDANA**

**SUBMITTED BY:**

|  |
| --- |
| **Hodard HAZWINAYO** |
| **Roll No: 202110036** |

1. **Problem Scenario**

**TASK: Given the dataset of social network users and their decisions for purchasing a car that was shown to them as an advertisement, develop a model that should predict the behavior of future users on after seeing the same advertisement.**

Marking: the models which be assigned marks based on their prediction scores and how the quality of the presentation such as motivating the choice of certain parameters.

1. **Data Set to be used.**



**How to choose machine learning model for the problem?**

several factors have impact on our decision to select best algorithm. Some problems are unique that specific algorithms are defines for these problems like recommendation system. Or in other cases these points can be helpful to decide best algorithm.

1. Study the data.
2. Understand the business problem.
3. Always keep in mind the constraints by the data or business.
4. Accuracy or speed, which really matters for the case.

**Solution**

Predicting product sales through ads delivered on Social Networking Sites using k-N.N. in Python



***In simpler words we tell whether a user on Social Networking site after clicking the ad’s displayed on the website, end’s up buying the product or not. This could be really helpful for the company selling the product.***

***Let’s say that it’s a car company which has paid the social networking site (For simplicity we’ll assume its Facebook from now on) to display ads of its newly launched car. Now since the company relies heavily on the success of its newly launched car it would leave no stone unturned while trying to advertise the car.***

***Well then what’s better than advertising it on the most popular platform right now. But what if we only advertise it to the correct crowd. This could help in boosting sales as we will be showing the ad of the car only to selected crowd.***

**K.Nearest Neighbors(k-N.N.)**

**A Gentle Introduction**

The k-NN algorithm is among the simplest of all machine learning algorithms. The input consists of the k closest training examples in the feature space while the output depends on whether k-NN is used for classification or regression:

In **k-NN classification**, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

In **k-NN regression**, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.

Well in this particular case we are dealing with a classification problem as we need to classify users as those who would but the car or not.

**HOW DOES THE ALGORITHM WORK?**

1. Choose the number K of neighbors
2. Take the K nearest neighbors of the new data point, according to Euclidean Distance
3. Among these K neighbors, count the number of data points in each category
4. Assign the new data point to the category where you counted the most neighbors.

## Part 1 — Data Preprocessing

# Importing the libraries

import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd

Importing the dataset

***The Dataset contains information about users on a Social Networking site and using that info as Features for our ML model, we try to predict that whether a particular user after clicking on an ad on the Social networking site goes on to buy a particular product or not.***

***Well this particular Social Network has a Business client which as mentioned earlier is a car company which advertises itself by putting adds on the social networking site. Now the work of the social network here is to gather information as to whether the user bought the product or not. The dependent variable in this case is Purchased which is 1 if user purchases the car and 0 otherwise.***

***So the goal here is to create a classifier which would put each user into the correct category by predicting as to whether he’s buying the product or not***.

dataset = pd.read\_csv('Social\_Network\_Ads.csv')

# printing the first few entries of the Dataset

print(dataset.head())

# Attention:

The following features will be considered as the independent variables…

**…1)Age**

**…2)Estimated Salary**

Now some of you might be wondering that the dataset also contains 3 more columns and why are we leaving them?

Well the answer to that is quite simple…and we will soon see the reason as to why each of them is being dropped.

**…1)UserId**- The **UserId** has no effect on whether the user would purchase the Car or not

**…2)Gender**- Some might say that **Gende**r would play a role but that is really subjective to discuss.

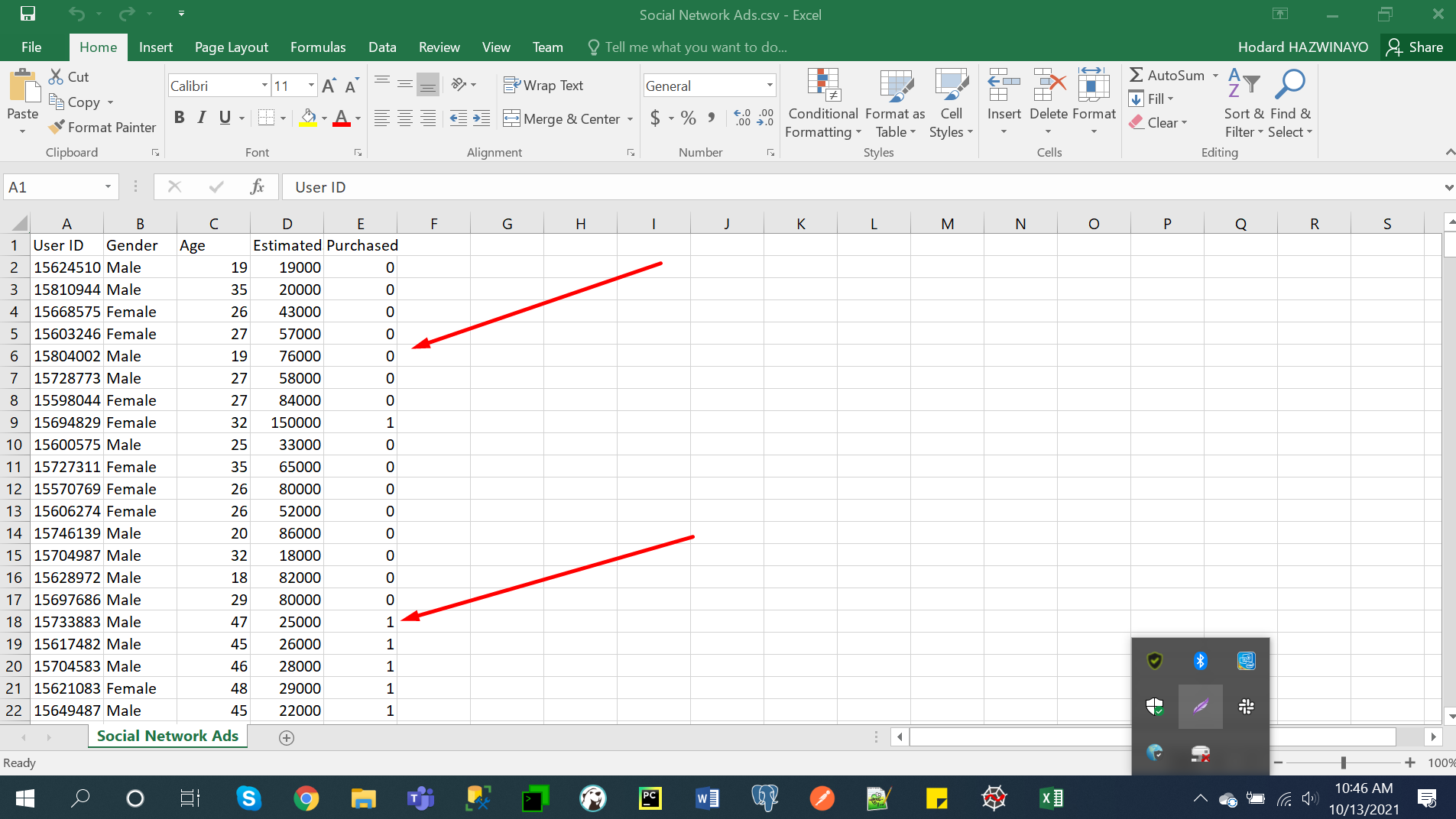
Moreover, since gender is a Categorical variable we would have to use Variable Encoder for it.

X = dataset.iloc[:, [2, 3]].values

Storing the dependent variable in y i.e. Purchased which is 1 if user purchases the car and 0 otherwise.

y = dataset.iloc[:, 4].values

This is what the Dataset actually looks lik…



**Splitting the dataset into the Training set and Test set**

Importing the Cross Validation library which is now known as ModelSelection in newer versions of Python

from sklearn.model\_selection import train\_test\_split

We divide the data into 75% data for training and 25% for testing our data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

Now are we going to apply Feature Scaling?

Yes, definitely we will be applying feature scaling because we want accurate prediction i.e. we want to predict which users are going to buy the car or not.

Feature Scaling

Importing Libraries

from sklearn.preprocessing import StandardScaler

Creating the standard Scalar Object of the Preprocessing Class

sc = StandardScaler()

Scaling X\_train by fitting the Standard Scalar object to our Matrix of Features X\_train

X\_train = sc.fit\_transform(X\_train)

Scaling X\_test in the same basis

X\_test = sc.transform(X\_test)

To actually see the difference and confirm that they are almost upto the same scale,if you want you can…

print(X\_train)

print(X\_test)

**Part 2 — Fitting our k-N.N. Model**

Fitting K-NN to the Training set

So we need to import the scikit.neighbours library and from it we would import the KNN Classifier

from sklearn.neighbors import KNeighborsClassifier

Creating an object of the class…

Inspect the classifier by pressing Ctrl+Q to show the Documentation and seeing all the parameters with their def accordingly

→No of nearest neighbors=5(Default)

→Specify metric as ‘minkowski’ and power as ‘2’ for using the Euclidean Distance for k-N.N.==> set p=2

classifier = KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2)

Now we fit the classifier object to our training set

classifier.fit(X\_train, y\_train)

**Part 3 — Predicting the Test set results**

Since the classifier has been fit to the Dataset we can predict the Outcomes of the test set.

y\_pred = classifier.predict(X\_test)

Displaying out the predicted values

print(y\_pred)

Now to calculate the accuracy of our model…

c=0

for i in range(0,len(y\_pred)):

if(y\_pred[i]==y\_test[i]):

c=c+1

accuracy=c/len(y\_pred)

print("Accuracy is")

print(accuracy)

So when you run this you get an accuracy of about 93% which is a great achievement for our classifier.

With this we end our predictions. Now the next section is of data visualization, which helps us visualize the accuracy and the errors of our model.

**Part-4 — Data Visualization and Confusion matrix**

Visualising the Training set results

from matplotlib.colors import ListedColormap

X\_set, y\_set = X\_train, y\_train

X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),

np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),

alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(X1.min(), X1.max())

plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y\_set)):

plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],

c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('K-NN (Training set)')

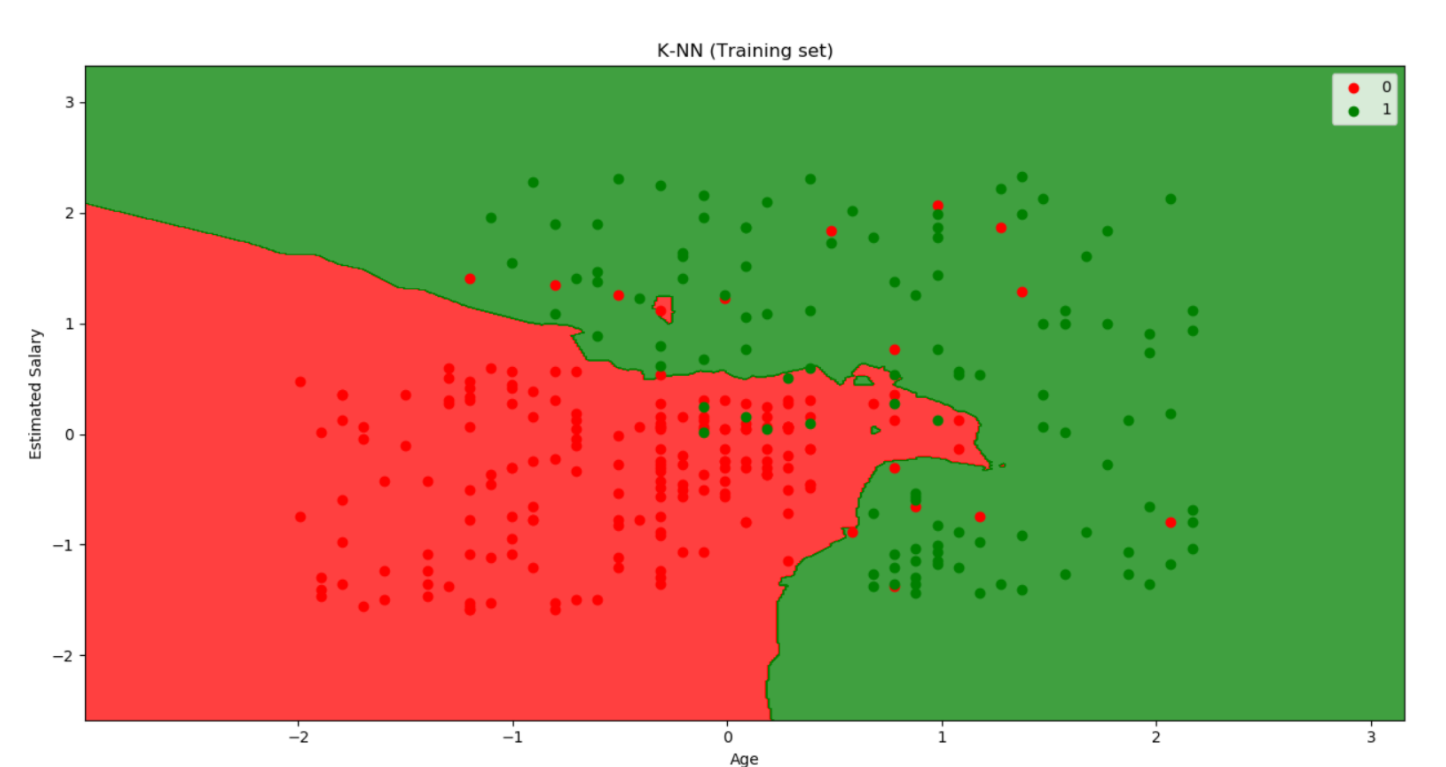
plt.xlabel('Age')

plt.ylabel('Estimated Salary')

plt.legend()

plt.show()

So now something like this would show after running the above code…



*As it might be visible in the newly opened graph of the Training Set that we have a Non-Linear Classifier which fits the Data pretty well.*

*Well apart from the very few misclassified points…Red points in Green region or vice versa our Model does a pretty decent job in classifying these points.*

**Visualizing the Test set results**

from matplotlib.colors import ListedColormap  
X\_set, y\_set = X\_train, y\_train  
X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 1, stop = X\_set[:, 0].max() + 1, step = 0.01),  
np.arange(start = X\_set[:, 1].min() - 1, stop = X\_set[:, 1].max() + 1, step = 0.01))  
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),  
alpha = 0.75, cmap = ListedColormap(('red', 'green')))  
plt.xlim(X1.min(), X1.max())  
plt.ylim(X2.min(), X2.max())  
for i, j in enumerate(np.unique(y\_set)):  
plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1],  
c = ListedColormap(('red', 'green'))(i), label = j)  
plt.title('K-NN (Training set)')  
plt.xlabel('Age')  
plt.ylabel('Estimated Salary')  
plt.legend()  
plt.show()



**Conclusion**

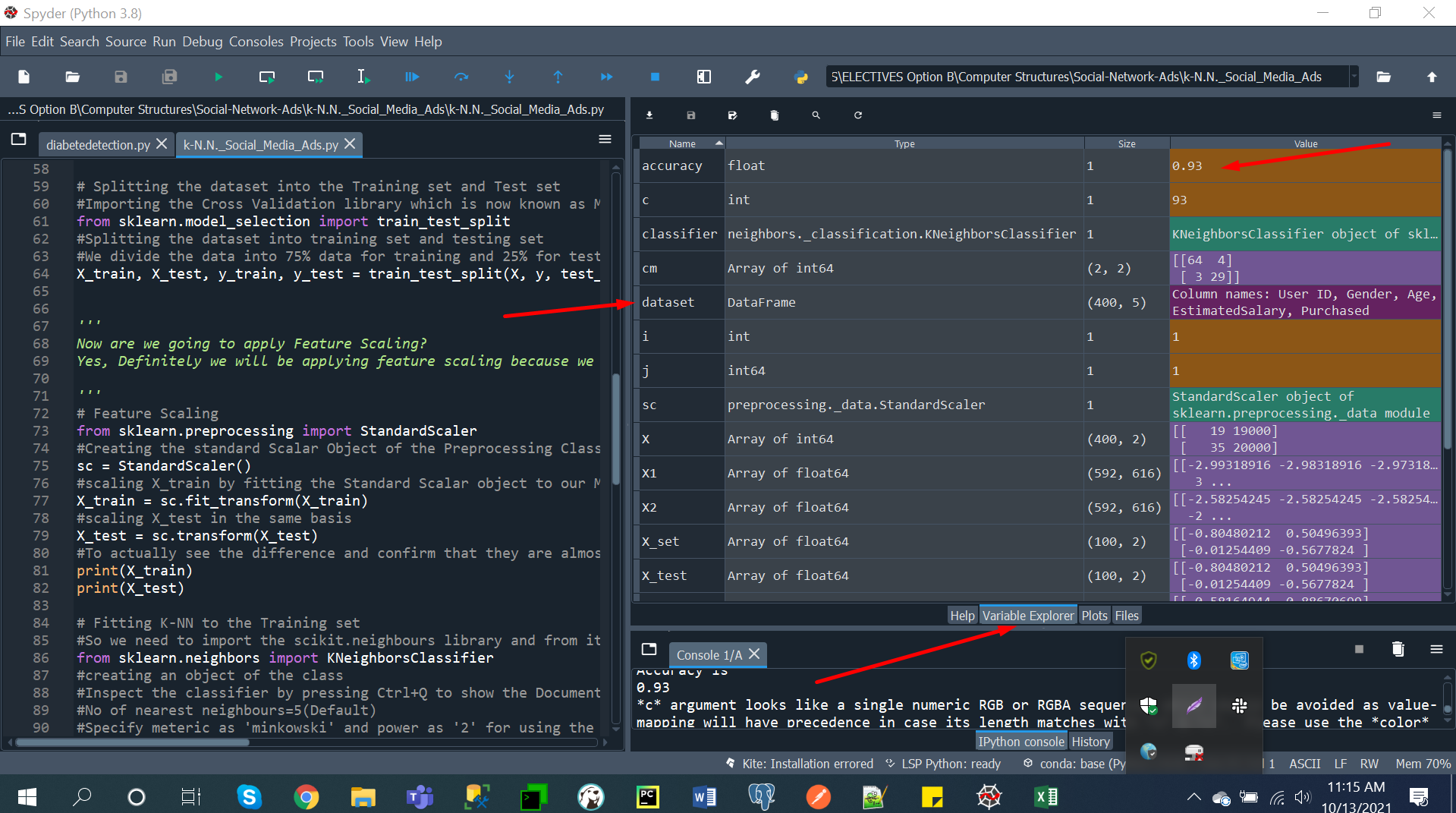
Just as we had performed an Analysis on the Graph of the Training set before we now perform one on the Test set results…

Yet Again we see that most of the points are correctly classified just with a few exceptions which is fine by the way to have, because we are trying to prevent our model from Over-fitting which we know can be a serious threat.

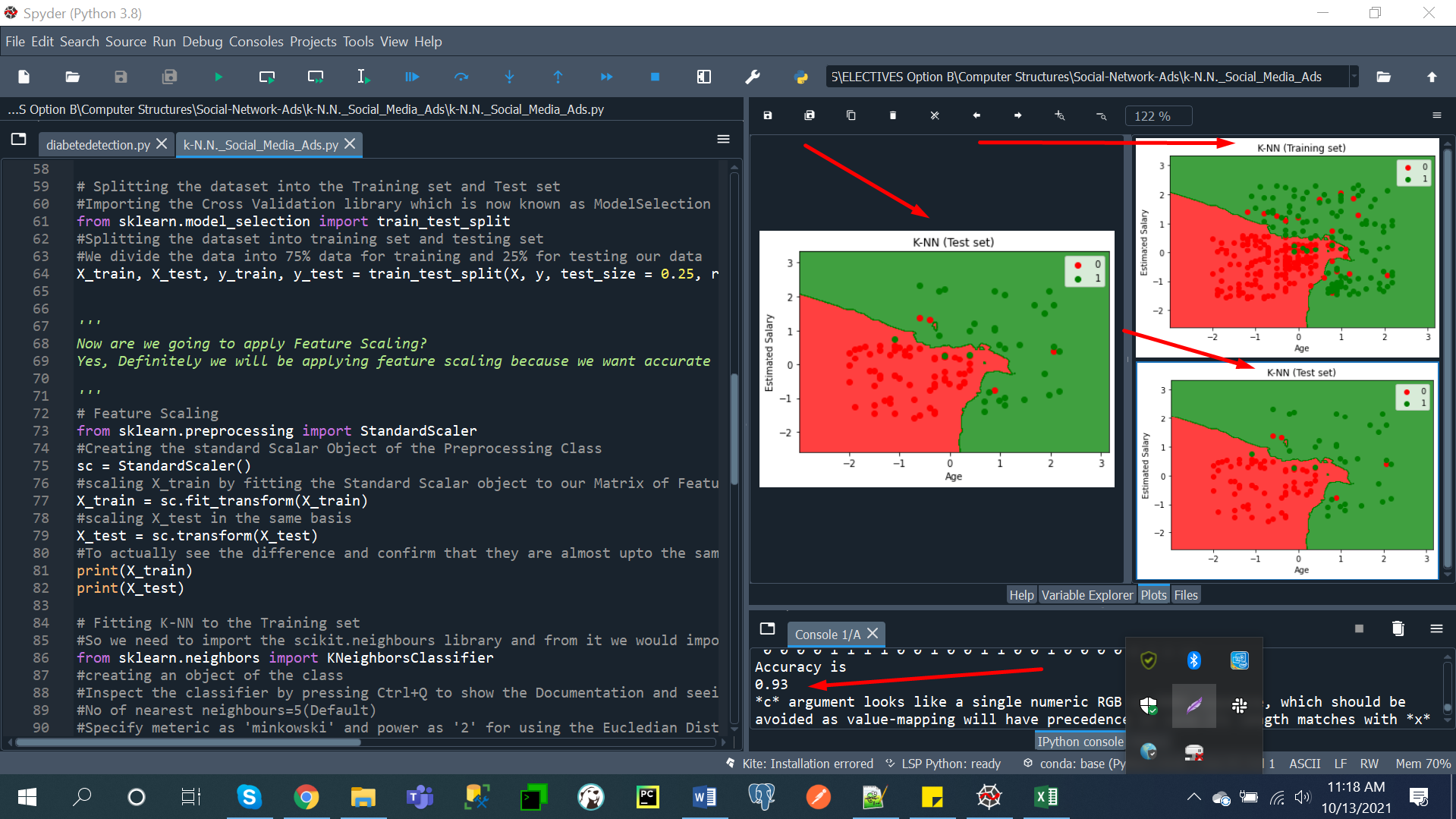
So now that we have even visualized the training and the test set graphs, we have officially completed building our model along with Data visualization.

Yayy!! So I just built a k-N.N. Model for predicting the sales of a product being advertised on a Social Media. The Car company which had hired someone as a Data Scientist would now be able to make a wise decision of targeting the correct crowd in order to advertise its brand new car and It would be the reason for its tremendous sales. Well I hope you are feeling proud already!!

**Variable Explorers menu on Spyder**



**Plots:**



**References:**

1. <https://www.kaggle.com/learn/intro-to-machine-learning>
2. <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>
3. <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>
4. <https://www.tutorialspoint.com/scikit_learn/scikit_learn_kneighbors_classifier.htm>
5. <https://machinelearningmastery.com/machine-learning-in-python-step-by-step/>
6. <https://rjunaidraza.medium.com/comparison-of-classification-algorithms-lr-dt-rf-svm-knn-6631493e300f>