Building a Predictive Model for Classifying Adult Income Levels



**By: Edward Ponce**

**Date: 10/12/2024**

# Table of Contents

[Table of Contents 2](#_Toc462280223)

[Main Objective 3](#_Toc2095356762)

[Data Description 3](#_Toc76771781)

[Preprocessing Object Data 4](#_Toc1813135067)

[Preprocessing Int64 Data 6](#_Toc568120139)

[Finalizing Preprocesss and Initializing the SVM 7](#_Toc682881611)

[Model Evaluation 8](#_Toc1038626509)

[Area Under Curve (AUC) Evalutation 9](#_Toc1448653947)

[Conclusion 10](#_Toc1504781190)

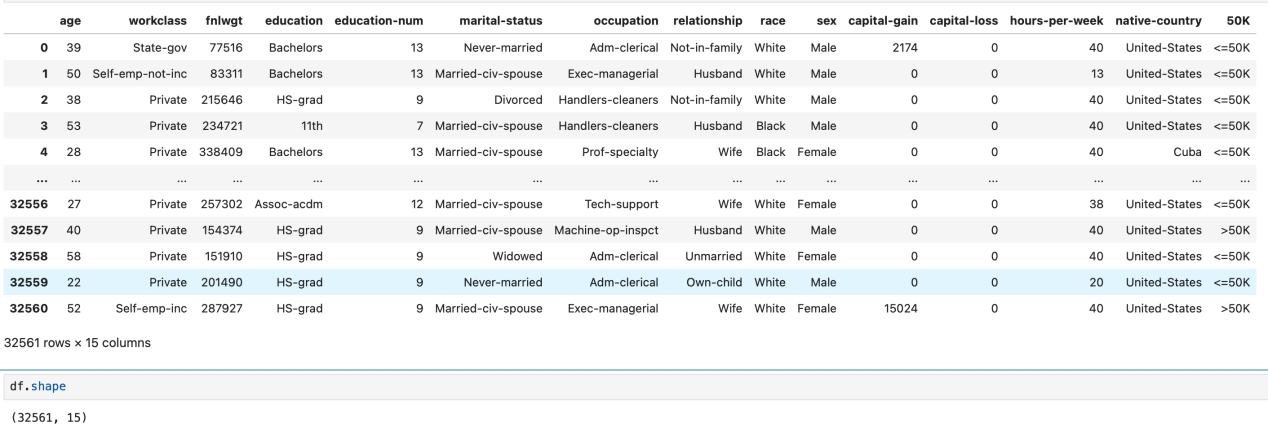
[Reflection and Improvement 10](#_Toc2090033258)

# Main Objective

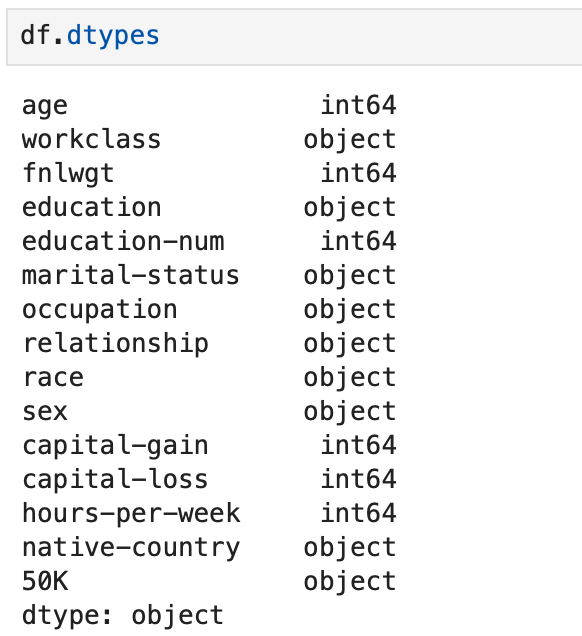
Given database adult.data we want to build a Support Vector Machine (SVM) to determine the whether an individual earns more than $50,000 per year or not. We are not interested in the overall interpretability of the model and therefore we are building the model with the primary purpose of predicting income levels to an acceptable degree defined by an AUC score above 0.8.

# Data Description

For this analysis we are given the database ‘adult.data’ with a shape of 15 columns and 32,561 rows.



Within this dataset, we have two primary datatypes: object and int64. Becuase we have object values in the dataset, we must consider encoding these values to guarantee that our SVM machine will correctly interpret each feature’s contribution to our predictive analysis.



The target in this data set will be the ‘50K’ column where the values are currently in type ‘object’ .

## Unknown

Before we begin disecting the data, we first want to understand the distribution of each feature and try to identify trends that might later inform us of the performance of our model.

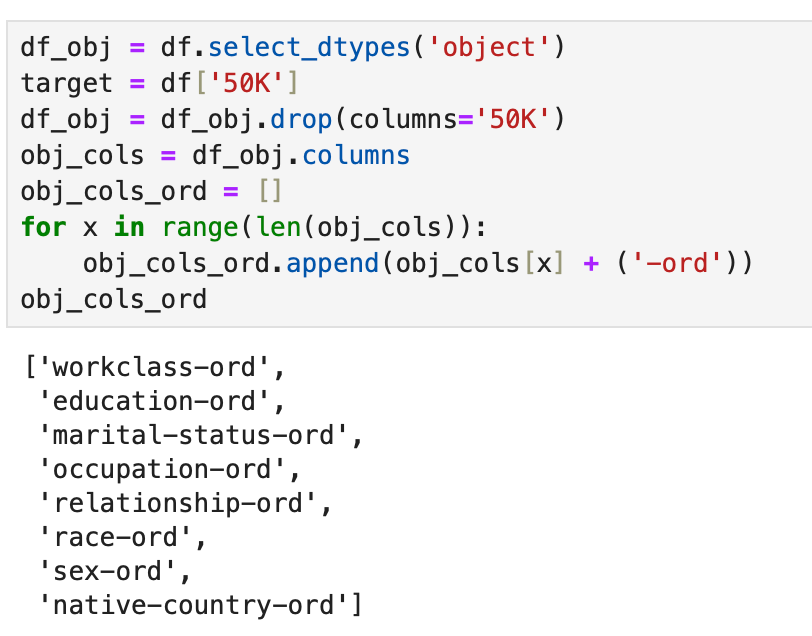
We can see in this feature pair plot, where those that earn more than $50k per year are highlighted in orange, that age and education are two very large factors in determing whethere someone earns over or under $50,000 per year.

## Preprocessing Object Data

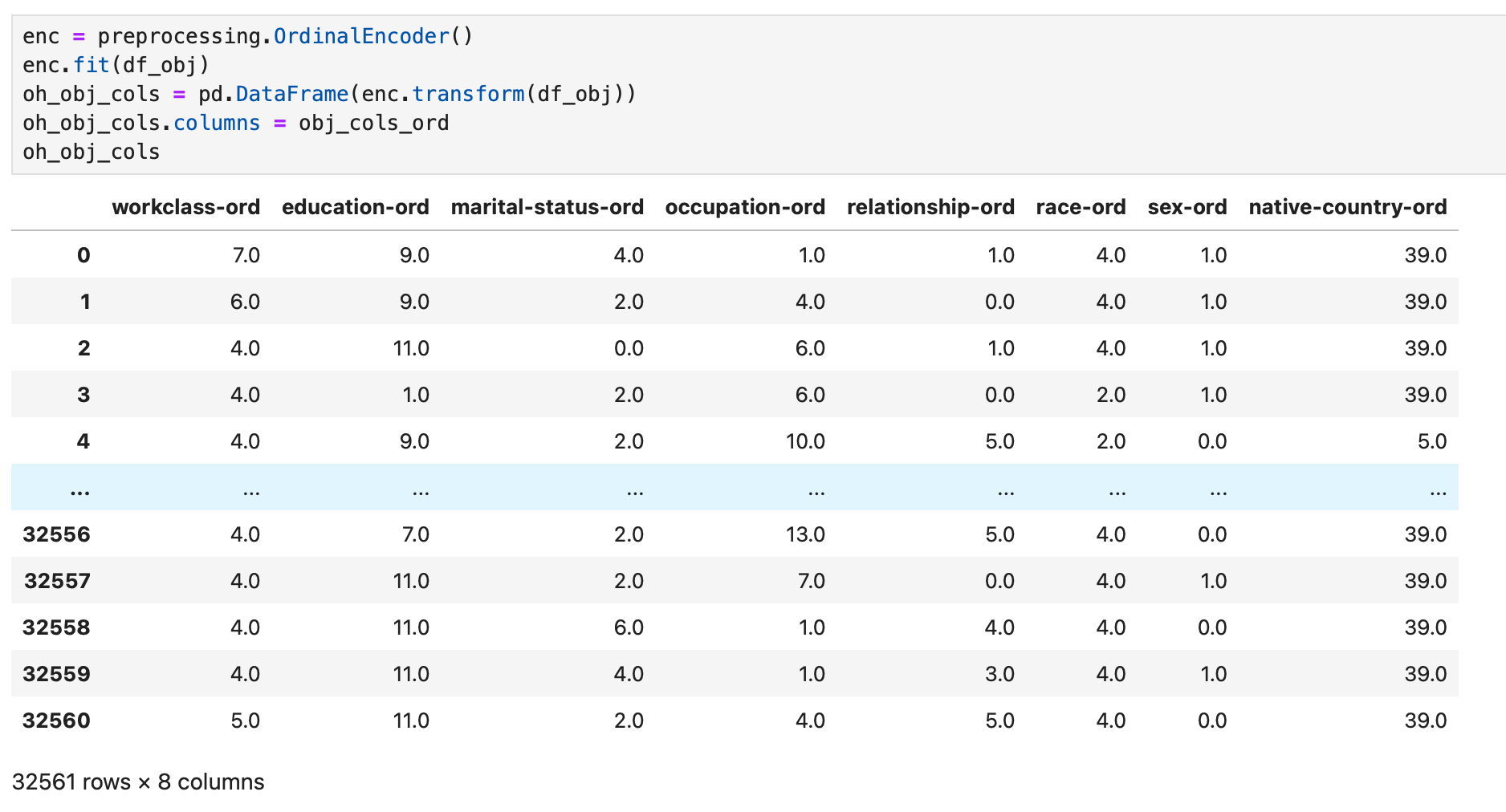
Before starting the analysis we must first preprocess the data to ensure there are no issues with the dataset. The first step to preprocessing the data is understanding if we have any missing values within the data.



We can see that there are no missing data points or errors in the dataset, thus we can move on to the next step of encoding the object values.



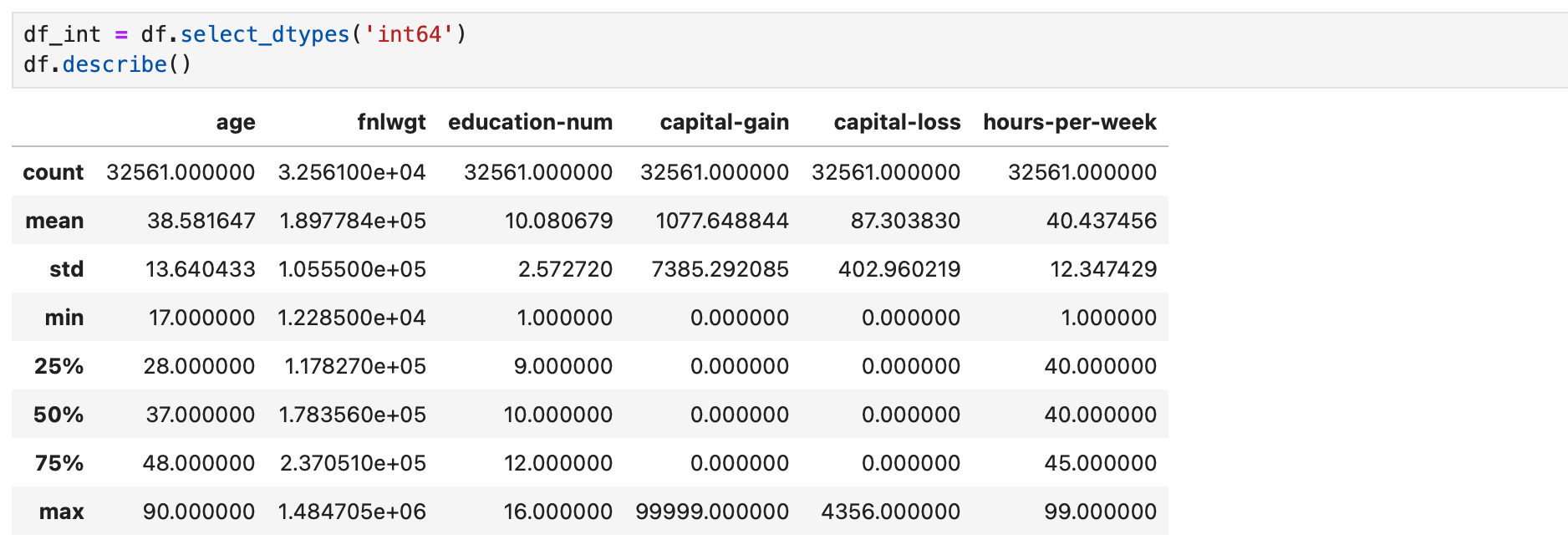
We begin the encoding process by only selecting columns of type ‘object’ and identifying the target column as the column ‘50K’ in the DataFrame ‘df’. After identifying this column, we drop it from the ‘object’ type dataset and begin to extract our feature names. It is important that we can identify the columns after being encoded and thus the original feature names are appending with the suffix ‘-ord’ to be assigned in the next step with the intention of identifying the ordinal encoded nature of the to-be preprocessed columns.



To begin the encoding process we initialize an **Ordinal Encoder** object from the **Sklearn** **preprocessing library.** This encoder is then fit only to the object data that does not include the target column. The previously generated column names are then assigned and used for future identifcation of ordinal encoded columns.

## Preprocessing Int64 Data

Now that the object data is ready we must focus on the 64bit integer data. Before any preprocessing we must describe and understand the data to inform what steps we must take to process this data. We should recall that none of the data entries were missing values and thus the statistical calculations provided by the ‘describe()’ function should accurately reflect the nature of the data.



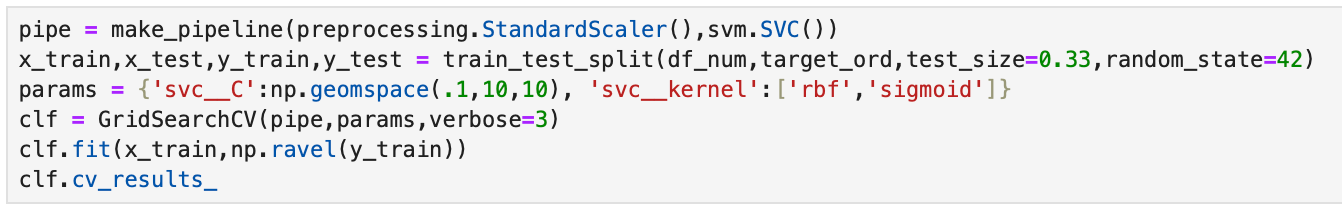
After describing the data we observe that there is a very large difference between min and max values within some of the integer features. There are also big differences in range of values from one feature to the next. This informs our decision that it is necessary to scale our integer data. Support Vector Machines are sensitive to numerical magnitudes and we do not want the values of features to inform the contribution of the feature to the model but rather have the variation and magintude within the feature inform this contribution.



Knowing that we should scale the data, we should first join the numerical ordinal data that we will be using for our SVM back into the data so that it too can be scaled. Note, the ordinal encoded data should have a min and max representing the numerical value asssigned to the first and last parameter of the feature. The column ‘50K’ has a min of 0 and a max of 1 because the unique values of the 50K feature were ‘<=50k’ and ‘>50K’ , identifying those who made less than or equal to $50,000/year and those who earned more than $50,000/year.

# Finalizing Preprocesss and Initializing the SVM

The Sklearn library provides many frameworks to streamline the processs of initializing and using machine learning tools, one of those being the ‘make\_pipeline’ function which allows us to create a queue of multiple tools to be executed.

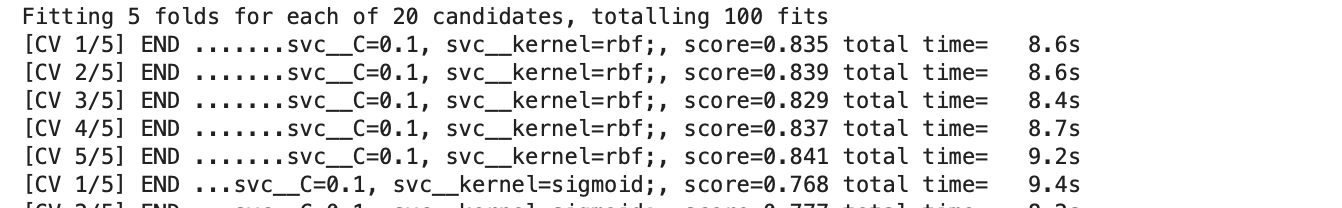


We inform the ‘make\_pipeline’ function that we would like to run the ‘StandardScaler()’ object of the preprocessing library, initialize our SVM, and use the return values of the ‘StandardScaler()’ function in our SVM operations.

Before beginning our fitting process we must first divide our dataset into training and testing. This is important because we want to be able to have a way to grade the performance of our machine and the only way to do this is to hide some of the data from it so that the machine isn’t trained on the answers to said tests. We choose to use 66% of the data for training purposes and 33% for testing purposes. This is a common split so that the machine has enough data to train but not too much such that we risk leaking and overfitting our model.

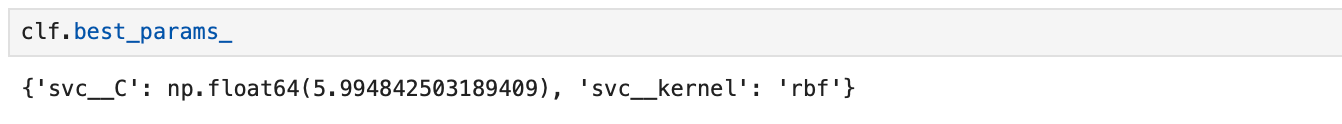
We want to be sure that the model is being trained and evaluated in such a way that we are finding the best possible outcome between various model parameters. To accomplish this, we are using GridSearchCV to executed a cross-value search for optimal parameters given a list of parameters to check. For the purposes of this analysis we have limited the parameters searched due to limitations in computation power and time. The first parameter chosen to be explored is the C, or regularization parameter, which affects the strength of regularization. An even distribution of 10 distinct values from .1 to 10 have been chosen for this parameter. The second parameter explored is the Kernel used, where only two were evaluated, Radial Bases Function and Sigmoid. By not using a linear kernel we can reduce our computation time by computing the similarity betwee pairs of data points to inform the classification of our data rather than having to transform the entire dataset.

These parameters alone result in 100 fits, 5 folds for each of the 20 candidates.

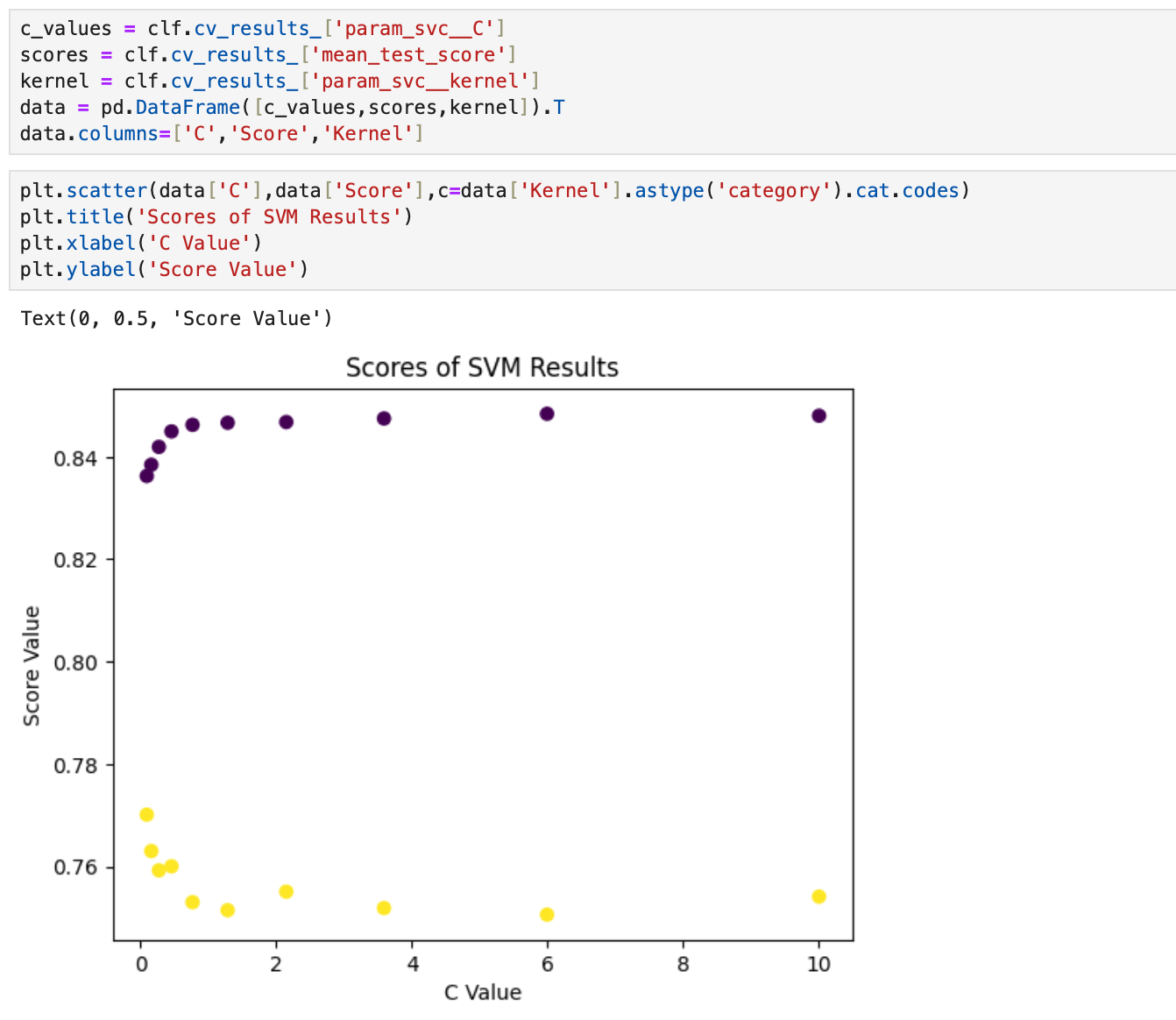


## Model Evaluation

After computing all variations provided by GridSearchCV given the list of our parameters, we arrive the best case of these parameters.

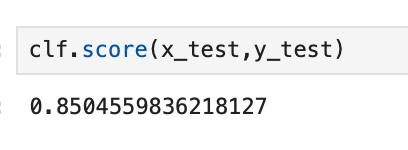


Our ideal regularization parameter is 5.994842503189409 using the Radial Basis Function.



To understand how the change in the regularization parameterss affected the accuracy of our model we plot C vs Score to obtain the scores of our kernels using different C values. RBF is colored in purple while Sigmoid is colored in yellow. We see that the sigmoid function had a reduction in score as the value of C increased. Meanwhile, the Radial Basis Function displayed the oppossite relationship where the score increased as the value of C increased. We should also note that while the score did increase as the value of C increased, the curve of the score flattened and dimishing returns lead to the conclusion to not seek further for a better value.

We can now test our model against our separated test data to see how it performs.



As expected, we see a similar score between our training and test datasets thus it is unlinkely the model is overfit to our training data.

## Area Under Curve (AUC) Evalutation

To further evaluate our model we plot the Area Under Curve diagram (AUC).



Our objective was to achieve an AUC value of more than or equal to 0.8 using our test dataset and after plotting the best estimator provided by GridSearchCV we come to a value of 0.89, greatly better than our objective.

In order to further understand if our model performed well, we use the Random Forest Classifier to benchmark our conclusion against. If we see a disparity in the AUC value, significantly higher or lower, we can begin to investigate if our model is underperforming or overperforming. In this case, our model slightly overperforms the Random Forest Classifier and thus this is an indicator that the model is performing as expected given the data provided.

## Conclusion

In this analysis we were able to arrive at a model that performed without our main objective using cross validation for a SVM machine with a changing kernel and regularization parameters. Before arriving at our conclusion we recognized that the data provided, although complete, demonstrated large ranges in values and different data types. We took this data and preprocessed appropriately given their data types, encoding the objects and scaling the integers. After this, we were able to compute our best estimator and arrive at a final configuration of a Regularization value of 5.99[...] while utilizing the ‘Radial Basis Function’ kernel. Our model was ultimately 85% accurate when tested against our test data set and provided a value of 0.89 in our AUC calculation, above the 0.8 in our main oobjective.

# Reflection and Improvement

If more computation power and time were available, we would likely be able to improve this moodel’s performance by cross-validating and using the full suite of parameters and kernels. We can also most likely generate a better model by combining several different models to leverage the benefits of different classification methods, however, this process would further complicate the analysis and potentially lead to a more severe ‘black box’ effect where interpretability would be entirely lost.