****

**Data Science Final Project**

|  |  |
| --- | --- |
| Member name | id |
| Ahmed Bilal | 29136 |
| Mauz Khan | 29590 |
| Habeel Wali | 29540 |
| Talha khan | 29366 |
| M Adeel Alarm | 29061 |
| Sufyian Abbasi | 29195 |

**Github link:** **https://github.com/Mauzkhan143/datascience**

**PROJECT REPORT**

1: Import Libraries

Import the usual libraries for pandas and plotting. You can import sklearn later on.

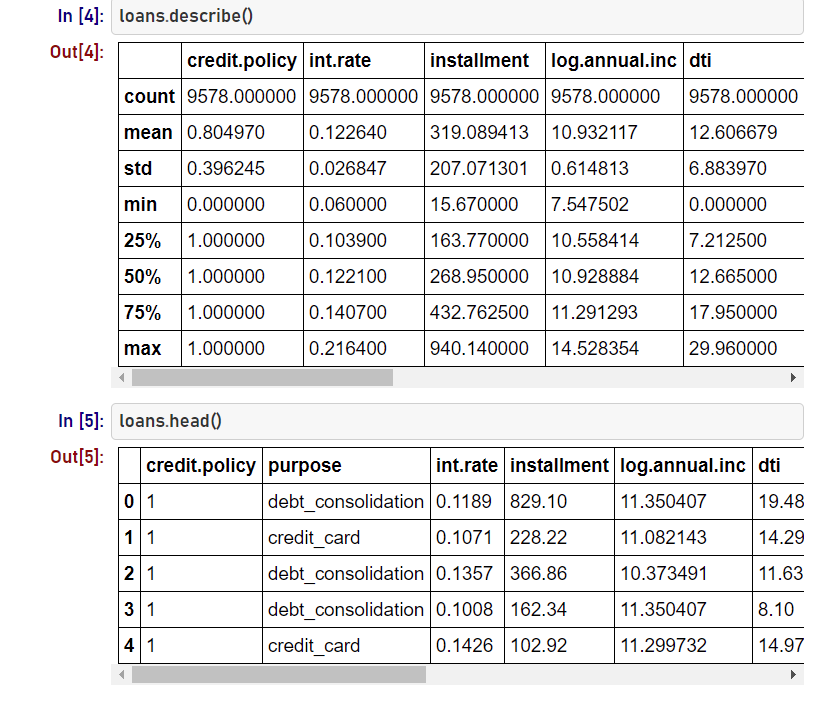
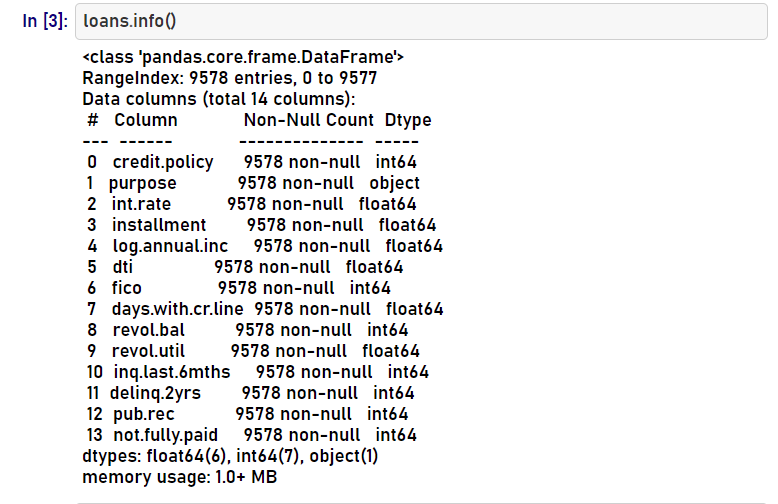


## 2: Get the Data

**Use pandas to read data\_loan.csv as a dataframe called loans.**



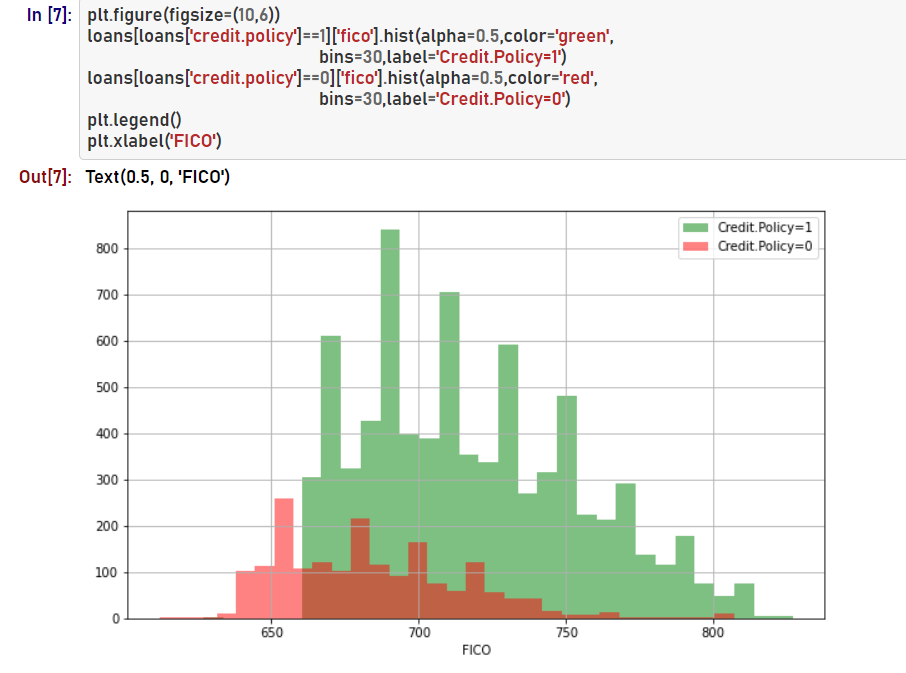
**Check out the info(), head(), and describe() methods on loans.**



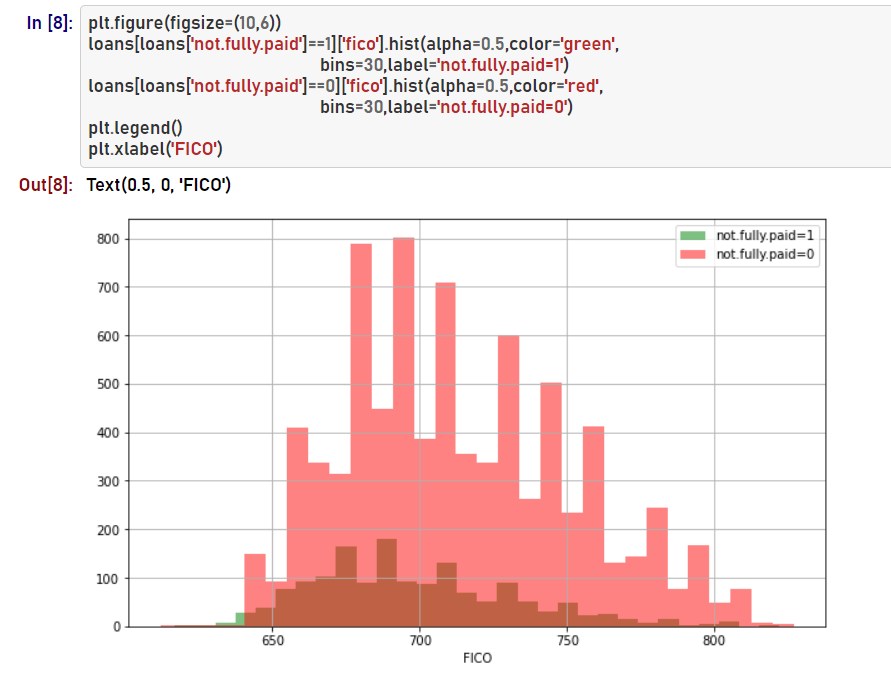
# 3: Exploratory Data Analysis

Let's do some data visualization! We'll use seaborn and pandas built-in plotting capabilities

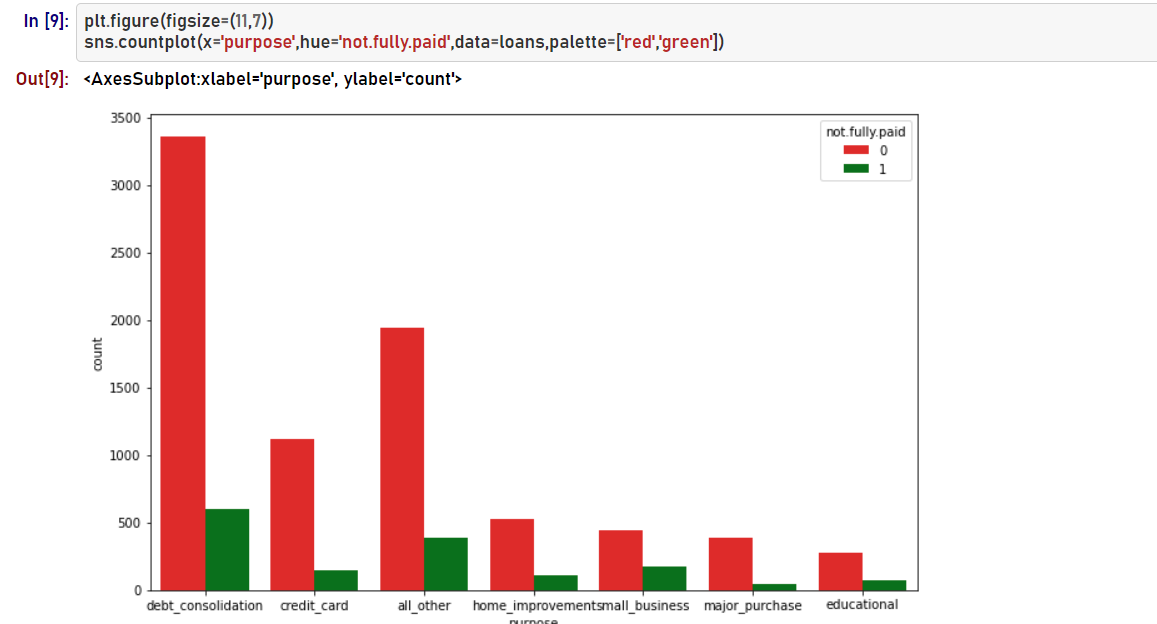
**Create a histogram of two FICO distributions on top of each other, one for each credit.policy outcome.**



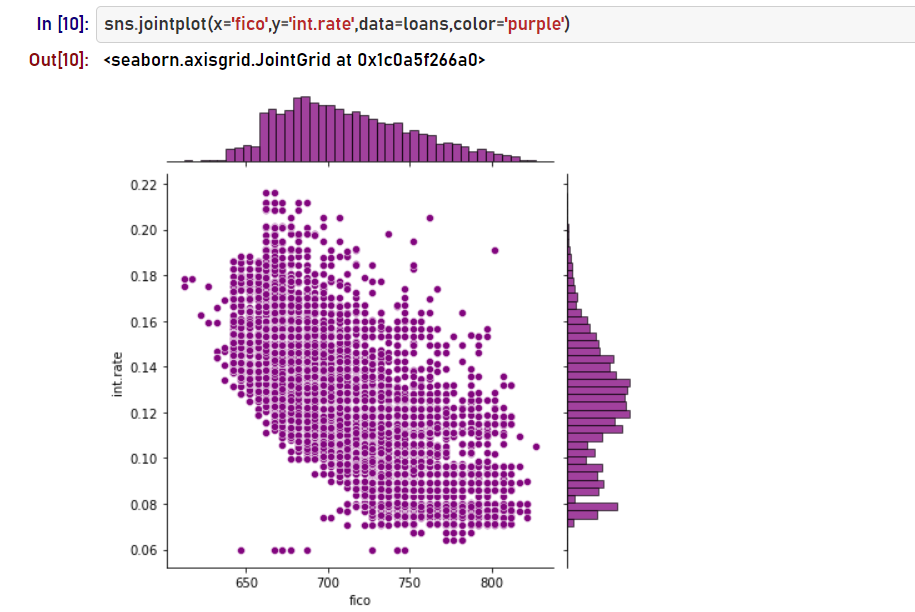
**Create a similar figure, except this time select by the not.fully.paid column.**



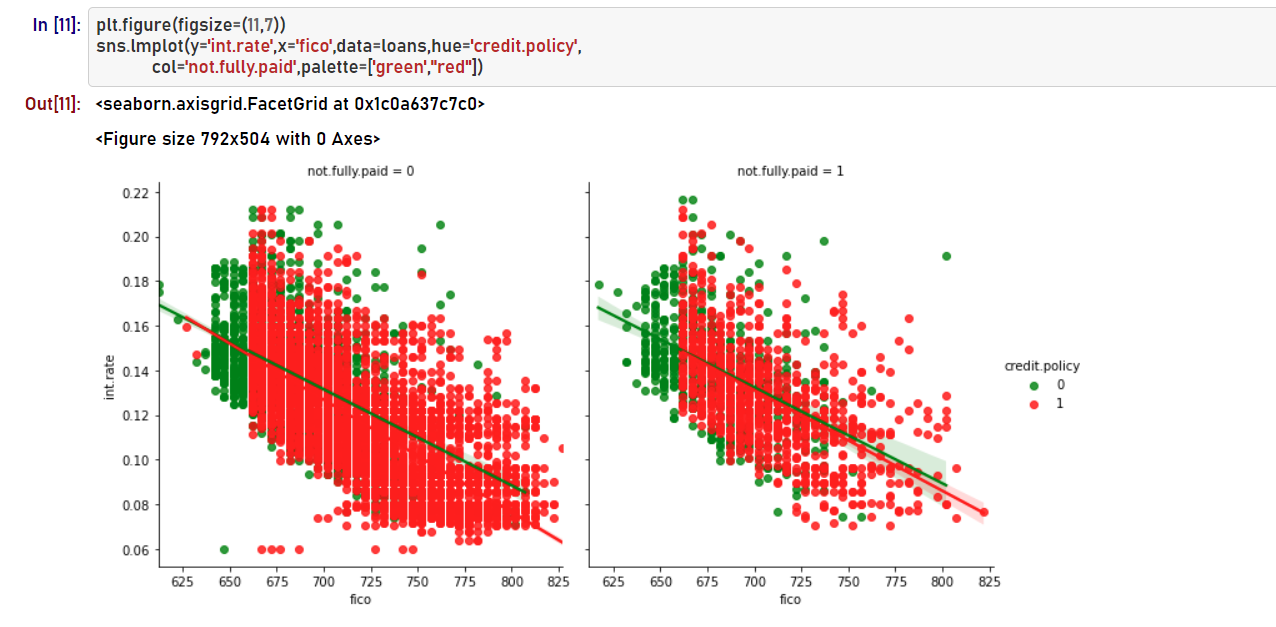
**Create a countplot using seaborn showing the counts of loans by purpose, with the color hue defined by not.fully.paid.**



**Let's see the trend between FICO score and interest rate. Recreate the following jointplot.**



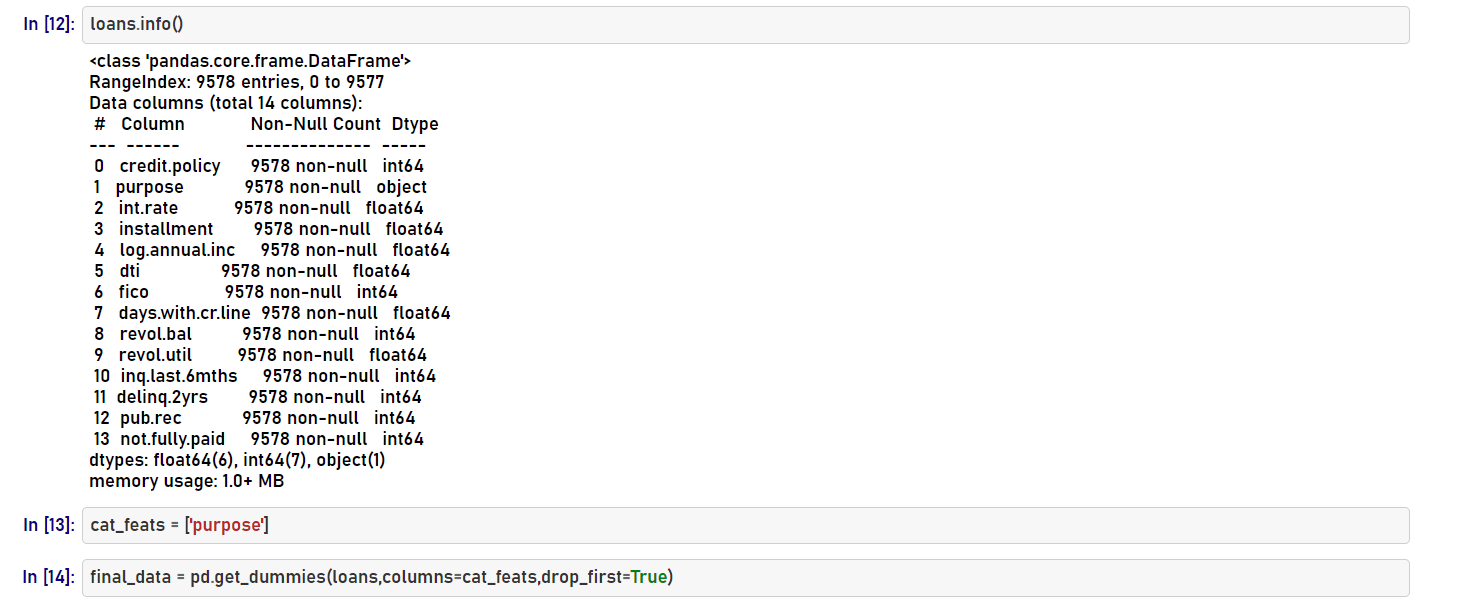
**Create the following lmplots to see if the trend differed between not.fully.paid and credit.policy. Check the documentation for lmplot() if you can't figure out how to separate it into columns.**



# 4: Setting up the Data[¶](https://render.githubusercontent.com/view/ipynb?color_mode=auto&commit=f212200c9de21223bf0529b871bdf94f4632ff9c&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f4f616b41636164656d792f4d616368696e652d4c6561726e696e672d776974682d507974686f6e2f663231323230306339646532313232336266303532396238373162646639346634363332666639632f4d414348494e455f4c4541524e494e472f4445434953494f4e5f54524545535f414e445f52414e444f4d5f464f524553545f414c474f524954484d2f4465636973696f6e5f54726565735f616e645f52616e646f6d5f466f72657374735f50726f6a6563745f536f6c7574696f6e732e6970796e62&nwo=OakAcademy%2FMachine-Learning-with-Python&path=MACHINE_LEARNING%2FDECISION_TREES_AND_RANDOM_FOREST_ALGORITHM%2FDecision_Trees_and_Random_Forests_Project_Solutions.ipynb&repository_id=343336575&repository_type=Repository#Episode-4:-Setting-up-the-Data)

Let's get ready to set up our data for our Random Forest Classification Model!

**Check loans.info() again.**

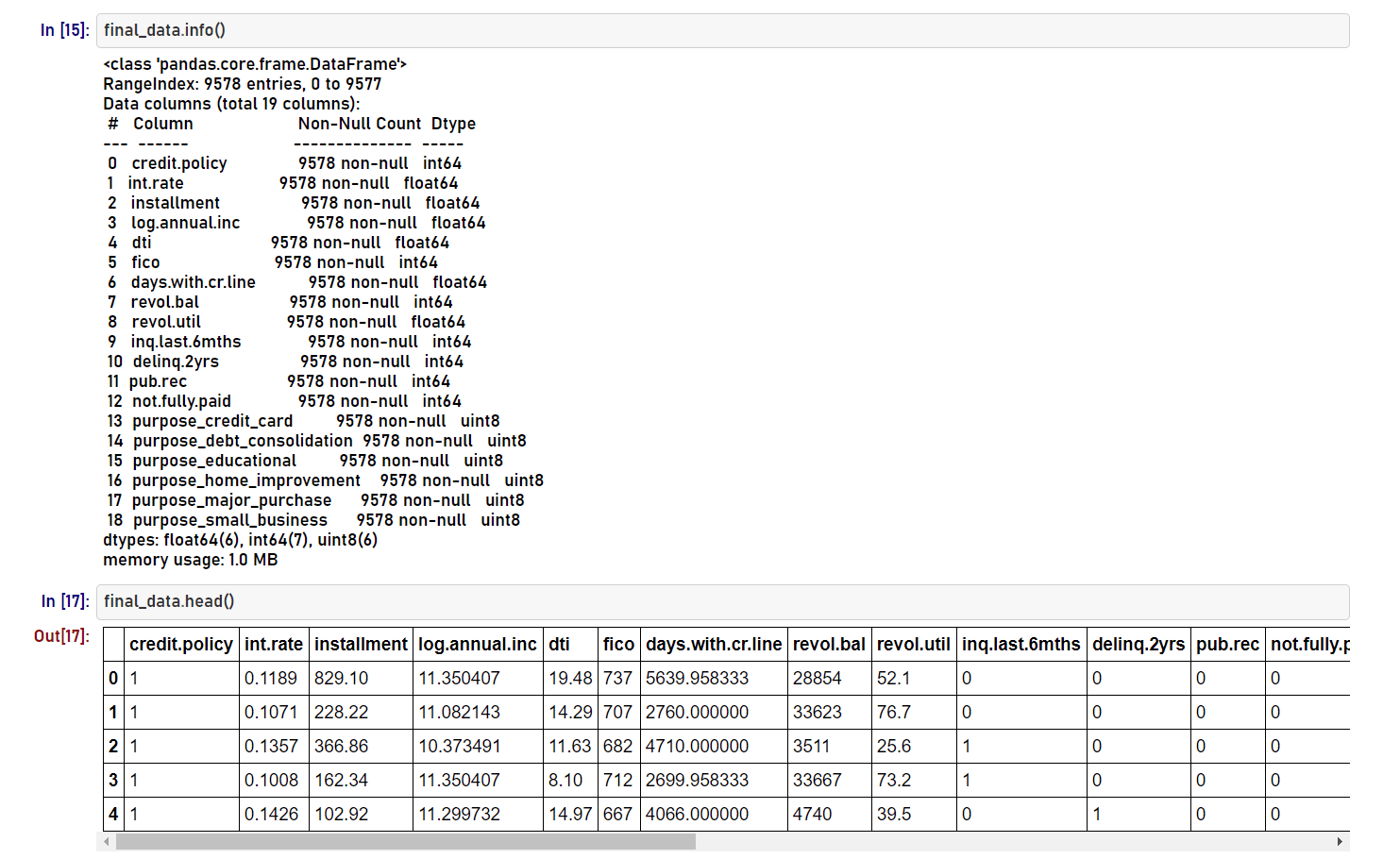


Notice that the **purpose** column as categorical

That means we need to transform them using dummy variables so sklearn will be able to understand them. Let's do this in one clean step using pd.get\_dummies.

Let's show you a way of dealing with these columns that can be expanded to multiple categorical features if necessary.

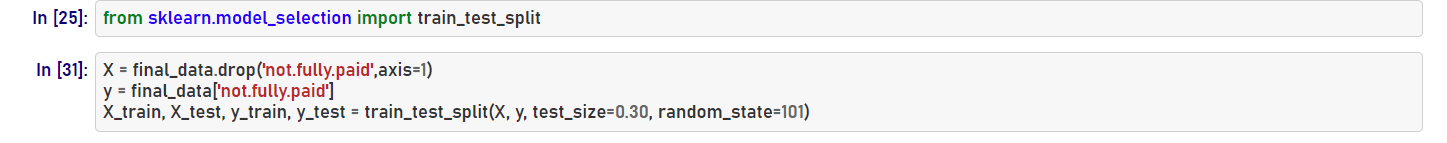
**Now use pd.get\_dummies(loans,columns=cat\_feats,drop\_first=True) to create a fixed larger dataframe that has new feature columns with dummy variables. Set this dataframe as final\_data.**



## 5: Train Test Split[¶](https://render.githubusercontent.com/view/ipynb?color_mode=auto&commit=f212200c9de21223bf0529b871bdf94f4632ff9c&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f4f616b41636164656d792f4d616368696e652d4c6561726e696e672d776974682d507974686f6e2f663231323230306339646532313232336266303532396238373162646639346634363332666639632f4d414348494e455f4c4541524e494e472f4445434953494f4e5f54524545535f414e445f52414e444f4d5f464f524553545f414c474f524954484d2f4465636973696f6e5f54726565735f616e645f52616e646f6d5f466f72657374735f50726f6a6563745f536f6c7574696f6e732e6970796e62&nwo=OakAcademy%2FMachine-Learning-with-Python&path=MACHINE_LEARNING%2FDECISION_TREES_AND_RANDOM_FOREST_ALGORITHM%2FDecision_Trees_and_Random_Forests_Project_Solutions.ipynb&repository_id=343336575&repository_type=Repository#Episode-5:-Train-Test-Split)

Now its time to split our data into a training set and a testing set!

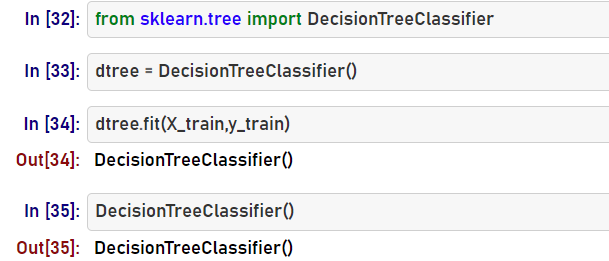
**Use sklearn to split your data into a training set and a testing set as we've done in the past.**



## 6: Training a Decision Tree Model[¶](https://render.githubusercontent.com/view/ipynb?color_mode=auto&commit=f212200c9de21223bf0529b871bdf94f4632ff9c&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f4f616b41636164656d792f4d616368696e652d4c6561726e696e672d776974682d507974686f6e2f663231323230306339646532313232336266303532396238373162646639346634363332666639632f4d414348494e455f4c4541524e494e472f4445434953494f4e5f54524545535f414e445f52414e444f4d5f464f524553545f414c474f524954484d2f4465636973696f6e5f54726565735f616e645f52616e646f6d5f466f72657374735f50726f6a6563745f536f6c7574696f6e732e6970796e62&nwo=OakAcademy%2FMachine-Learning-with-Python&path=MACHINE_LEARNING%2FDECISION_TREES_AND_RANDOM_FOREST_ALGORITHM%2FDecision_Trees_and_Random_Forests_Project_Solutions.ipynb&repository_id=343336575&repository_type=Repository#Episode-6:-Training-a-Decision-Tree-Model)

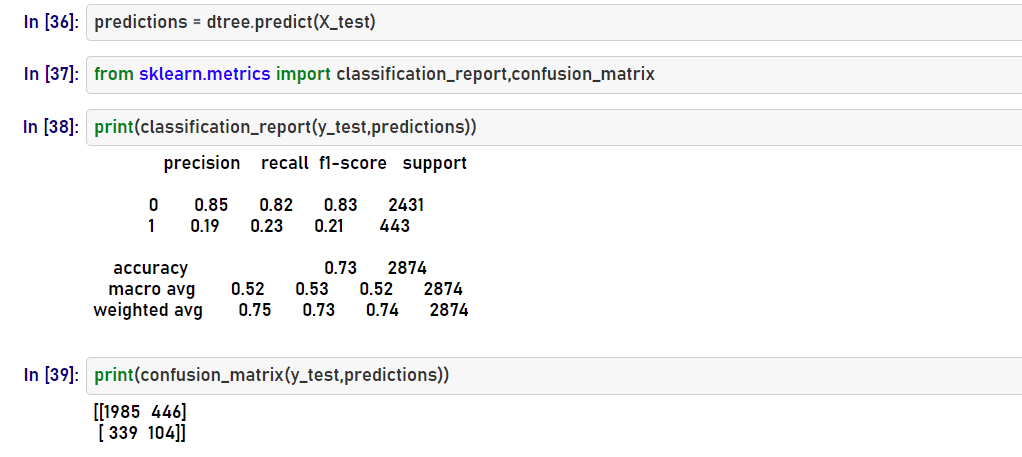
Let's start by training a single decision tree first!

**Import DecisionTreeClassifier**



## Predictions and Evaluation of Decision Tree

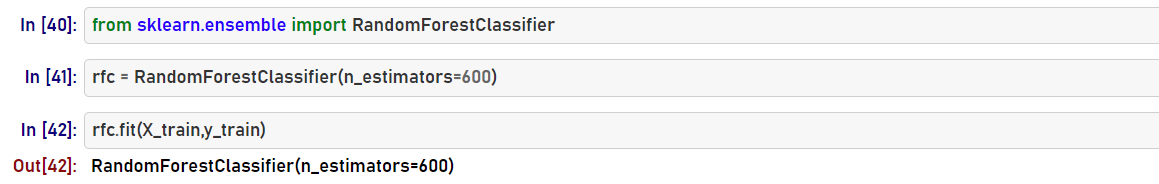
**Create predictions from the test set and create a classification report and a confusion matrix.**



## 7: Training the Random Forest model

Now its time to train our model!

**Create an instance of the RandomForestClassifier class and fit it to our training data from the previous step.**



## Predictions and Evaluation

Let's predict off the y\_test values and evaluate our model.

**Predict the class of not.fully.paid for the X\_test data.**

**Now create a classification report from the results. Do you get anything strange or some sort of warning?**

