When to Buy Airline Tickets

User Tutorial

IT 7993: Capstone

James Pock

Bijay Tamang

Angel Salcedo

Mohan Khadka

Contents

Contents

[Overview 3](#_1fob9te)

[Software Needed 4](#_3znysh7)

[Source Code & Datasets 5](#_2et92p0)

[Starting Anaconda and Jupyter Notebook 6](#_tyjcwt)

[Loading the Source Code 8](#_3dy6vkm)

[Running the Source Code 9](#_1t3h5sf)

[Loading Python Packages 9](#_4d34og8)

[Loading the Dataset 10](#_2s8eyo1)

[Displaying Graphs 11](#_17dp8vu)

[Analyzing the Datasets 12](#_3rdcrjn)

[Set the Columns for X and Y 13](#_26in1rg)

[Splitting Data into Training and Test Sets 14](#_lnxbz9)

[Running the Regression Models 15](#_35nkun2)

[Predictive Model GUI 17](#_1ksv4uv)

[Running the GUI 17](#_44sinio)

[Modifying Model for the GUI 20](#_2jxsxqh)

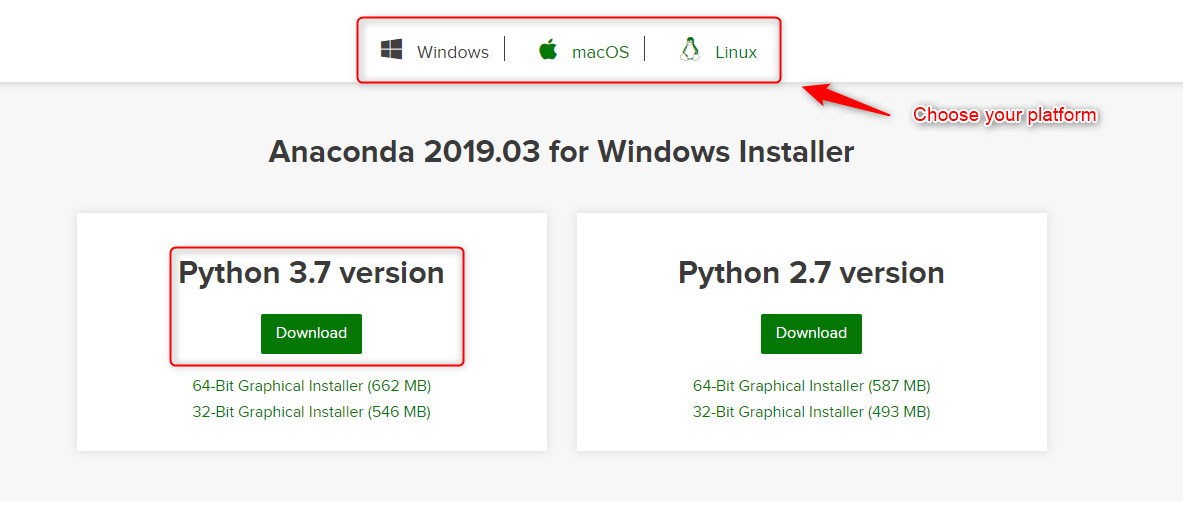
# Overview

The goal of this project is to find the best time to buy an airline ticket. Airline ticket prices vary depending on how soon the flight is. Using attributes of a specific flight, the team will try to predict the optimal date to purchase a ticket. For instance, it is generally true that the farther in advance you buy a ticket the cheaper it will be. However, this is a very simple heuristic which may not be true. In this project, the team will be in charge with determining the relationship between purchase date and flight cost. The purpose of this user tutorial is to provide instructions on how to recreate the predictive model that the team was able to create. This document will also help the user recreate the results we deemed valuable for this research.

# Software Needed

Anaconda Python 3.7 or later: <https://www.anaconda.com/distribution/#download-section>

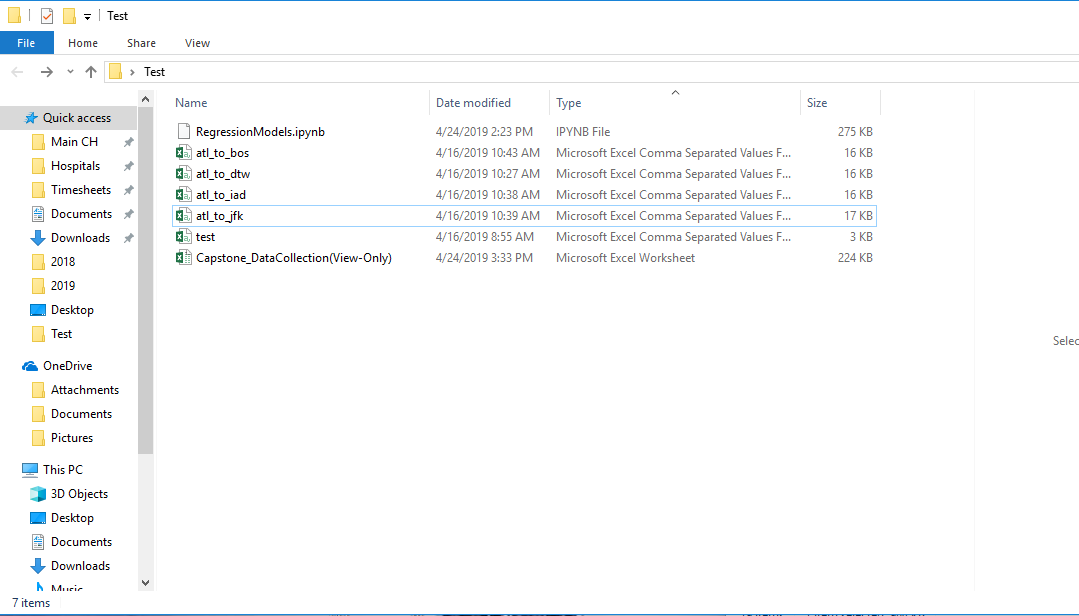
Download for the link provided and install. It may take a few minutes.



# Source Code & Datasets

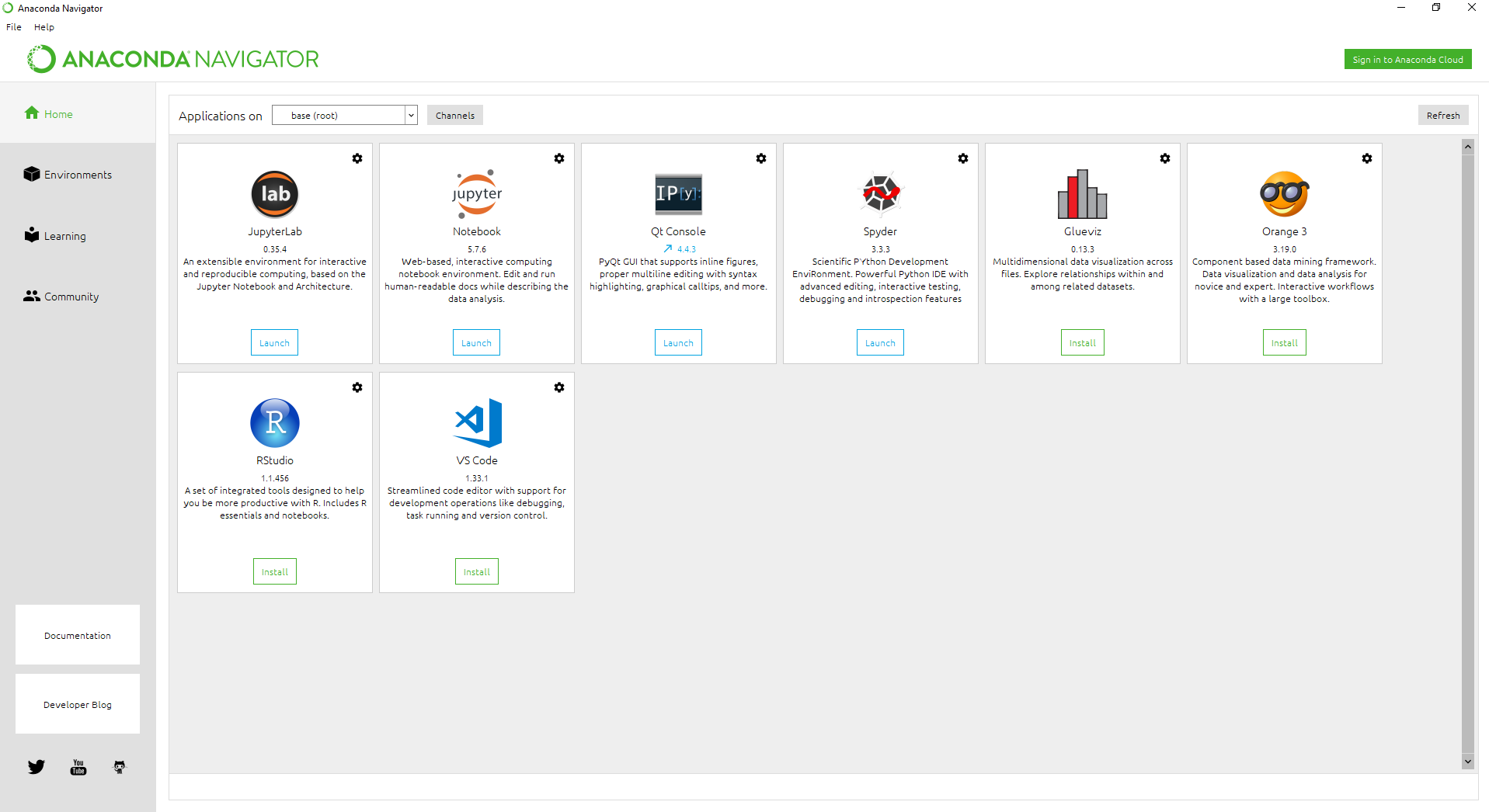
GitHub: <https://github.com/bijaytmg/capstone-project-airline>

* Download the folder from the GitHub link above. This folder will include all source code and datasets.
* Create a folder on your desktop called “Test”
* Unzipped the folder you downloaded and extract the contents to the “Test” folder you just created.

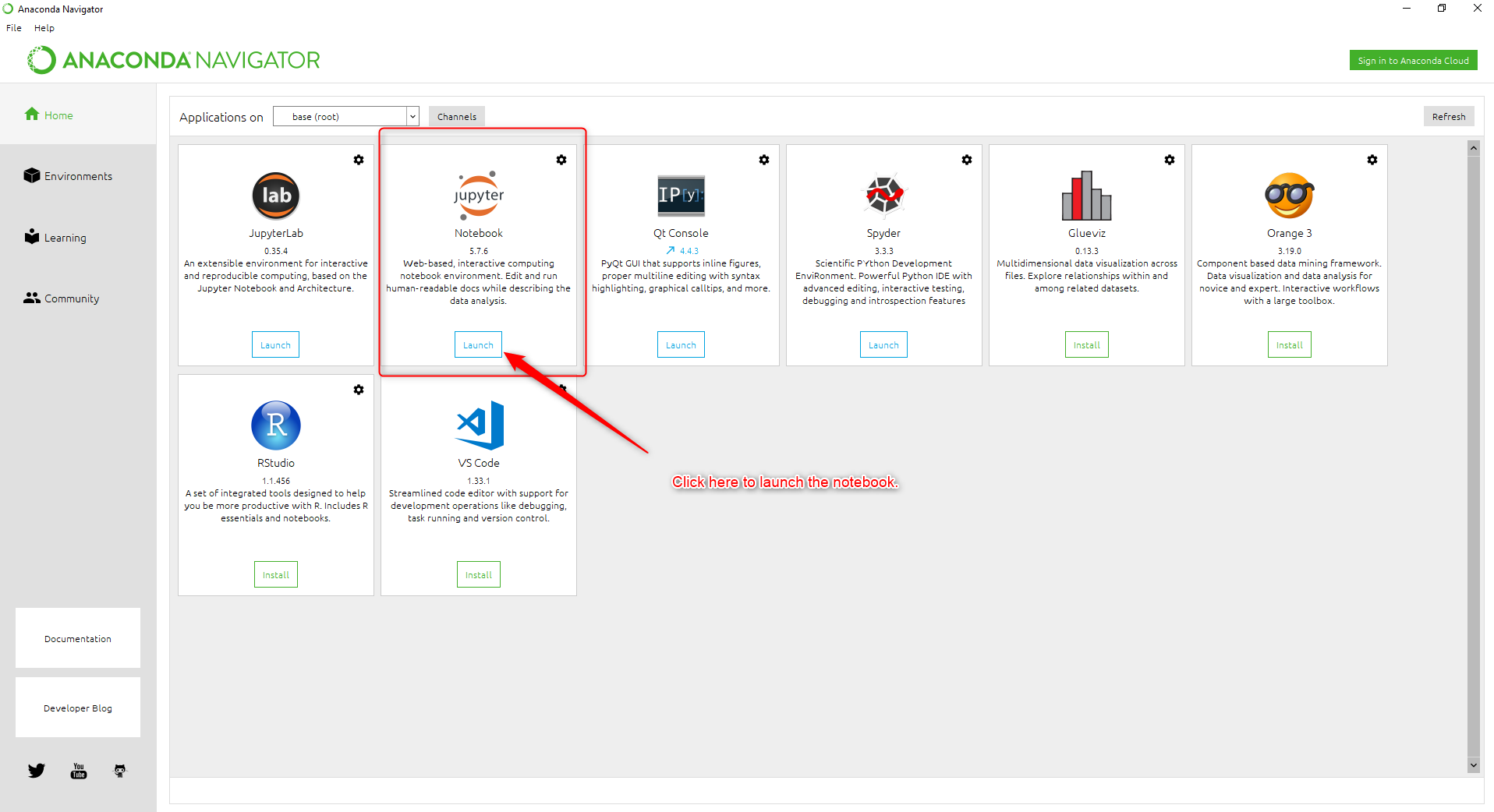


# Starting Anaconda and Jupyter Notebook

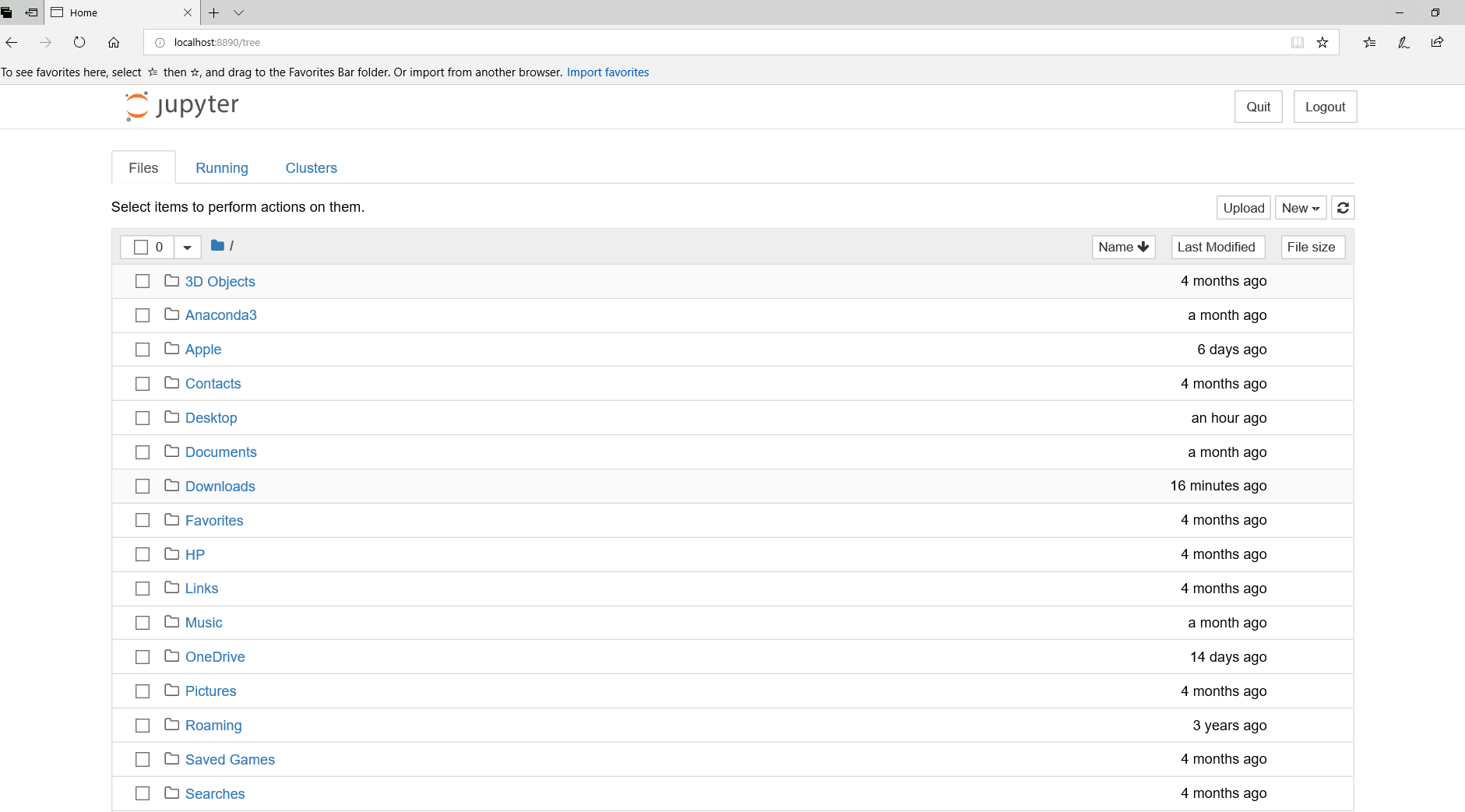
Once Anaconda finish installing, start the program. You should see an interface like this:



To start using Python code, you will need to launch Jupyter Notebook from the Anaconda Interface:

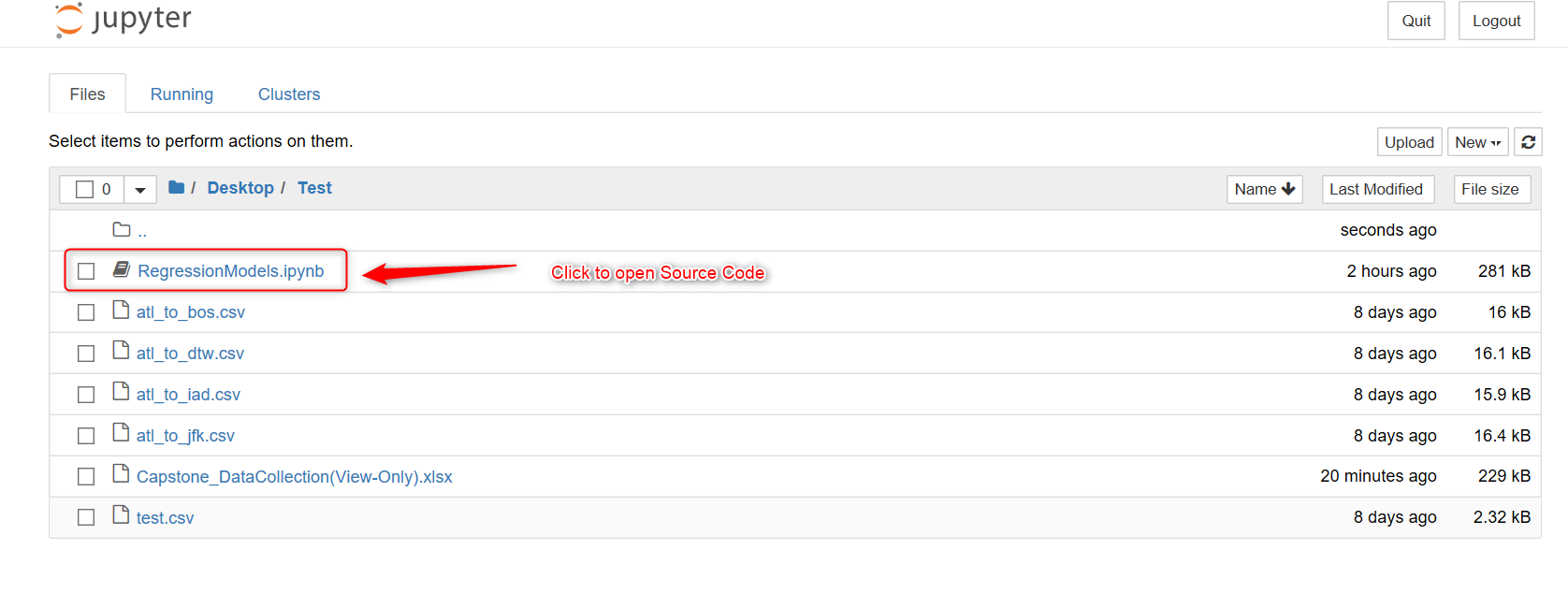


A web browser will open depending on your system’s default. You should see something similar like this. It will have your computer’s directories mapped out.

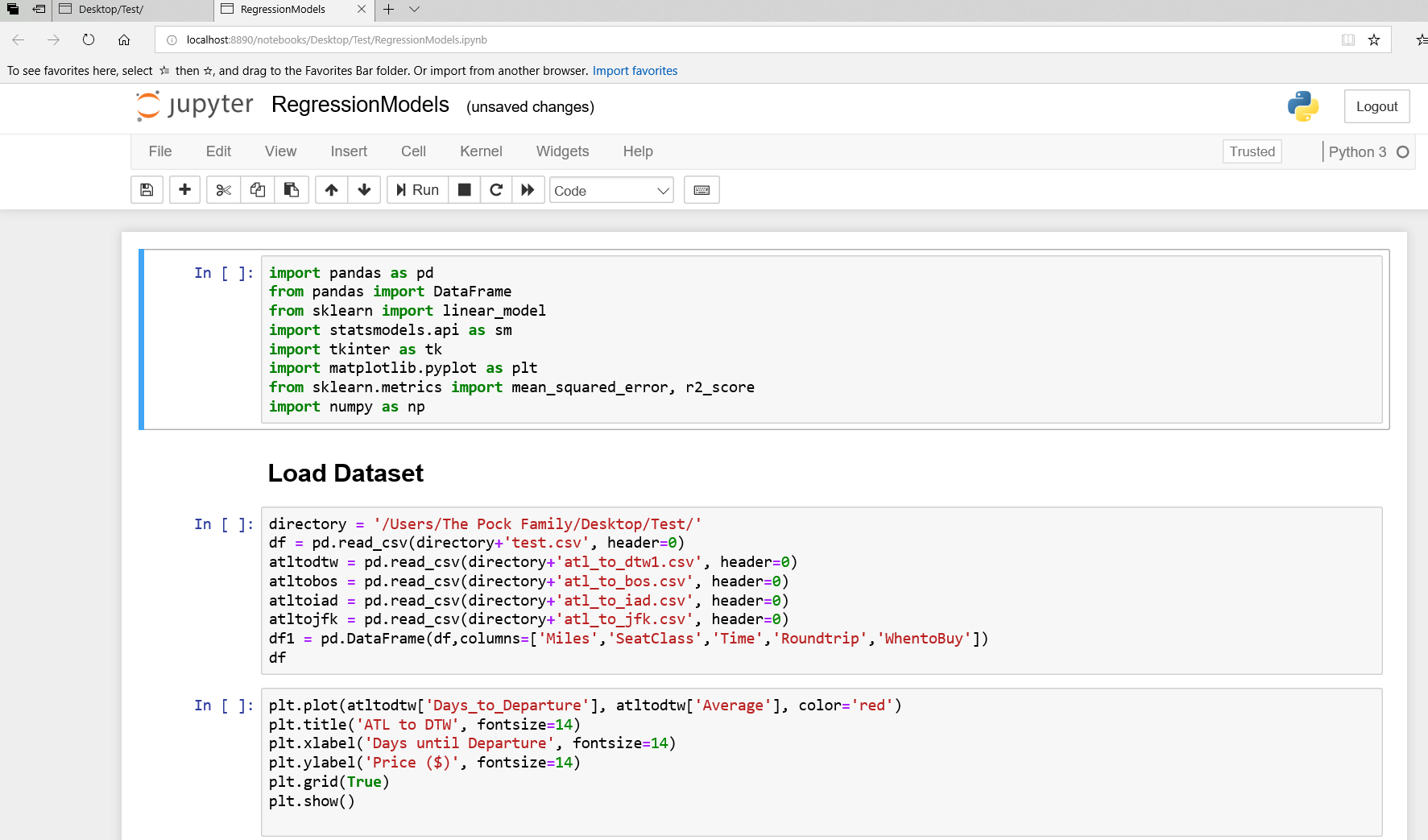


# Loading the Source Code

To load the source code, navigate to the “Test” folder you created earlier. You should be able to see all the files that you downloaded. **Click on RegressionModels.ipynb** to open the source code.



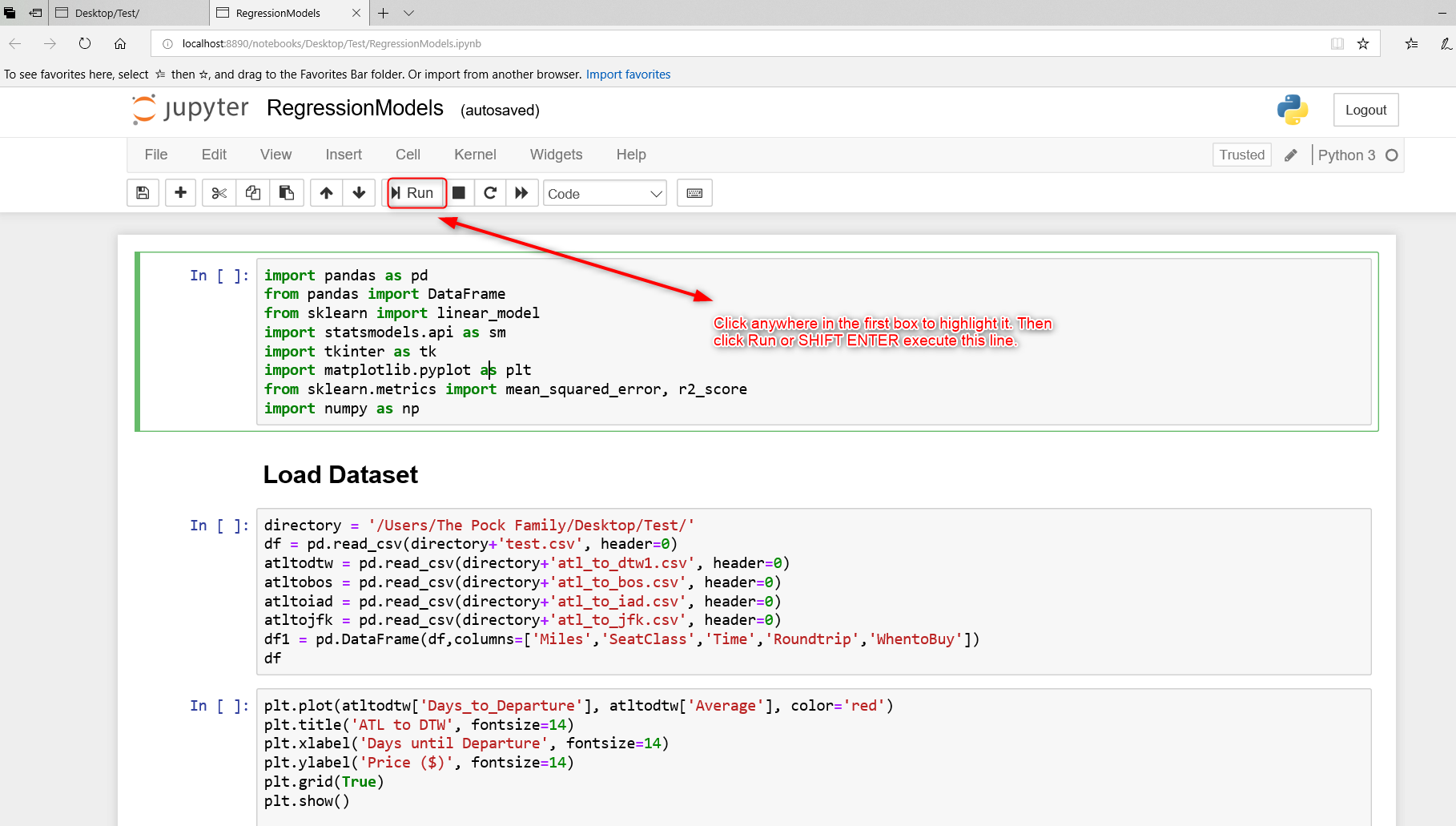
Once you click on the source code file, a new tab in your browser will open. The source code will displayed.



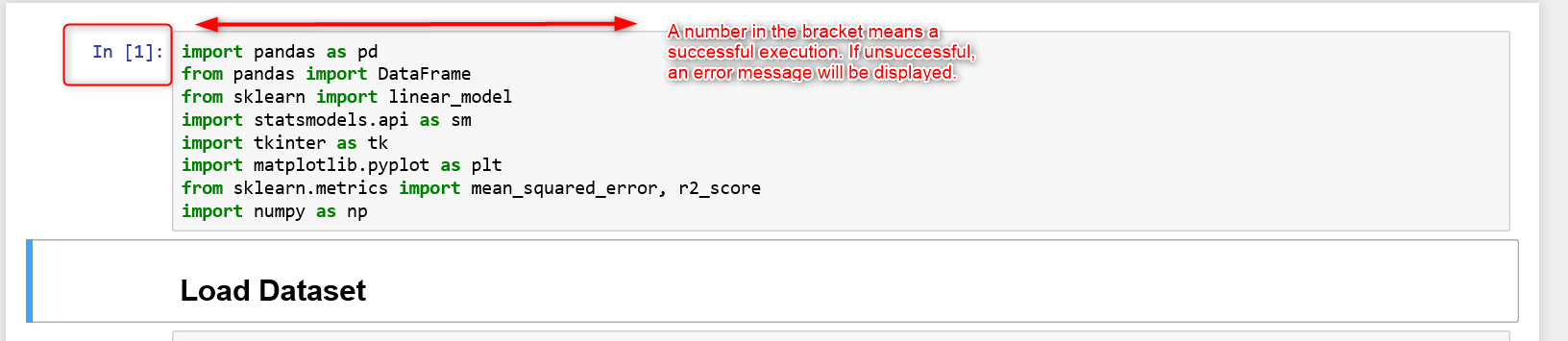
# Running the Source Code

## Loading Python Packages

The first box is very important as it contains the packages you will need to execute the source code. Be sure to run this first. Click anywhere in the first box. This will highlight the first box with a green border. To run this code, you can click “Run” or use your keyboard by pressing SHIFT ENTER.



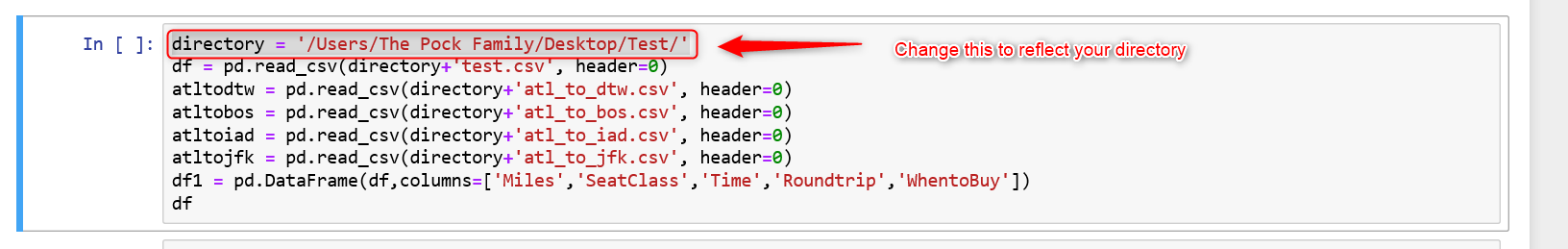
If successful, a number will appear in the bracket to the left of code. The number “1” should appear indicating this is the first line of code to execute successfully.



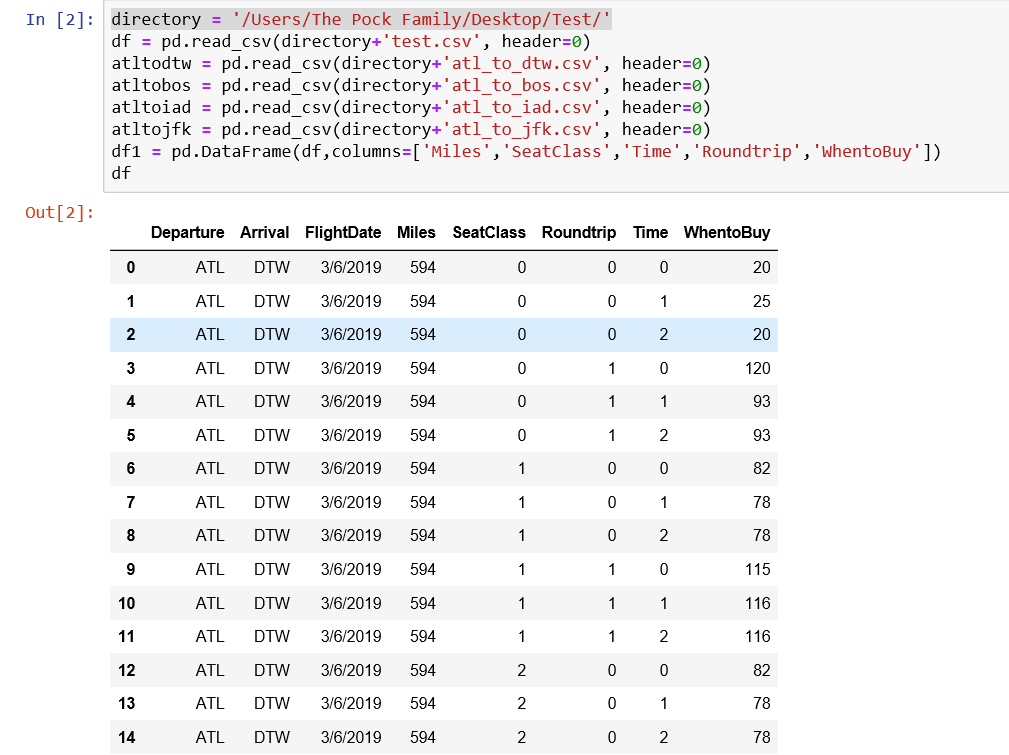
## Loading the Dataset

The next line of code will require the user to change the directory. Currently, it is defaulted to the project’s team computer.

Change “directory = '/Users/The Pock Family/Desktop/Test/'” to reflect your directory that contains the Test folder.

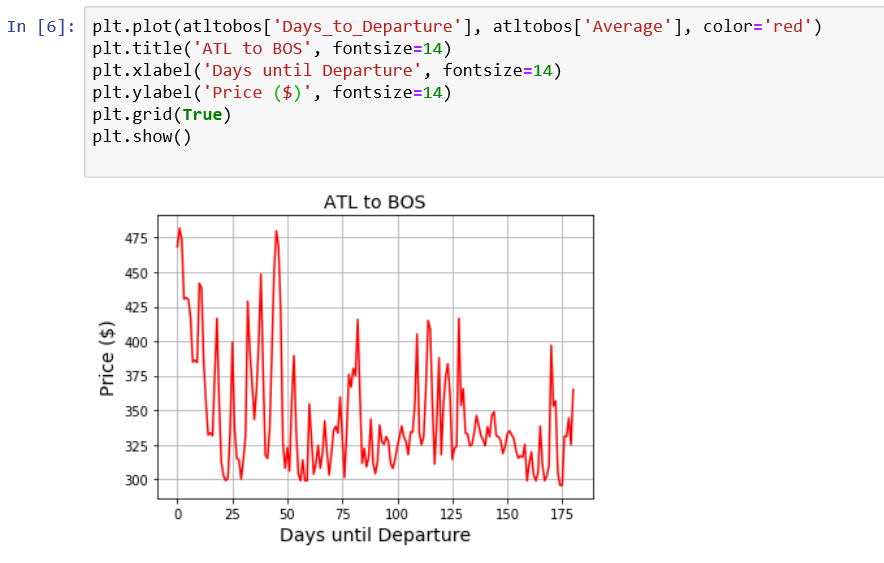


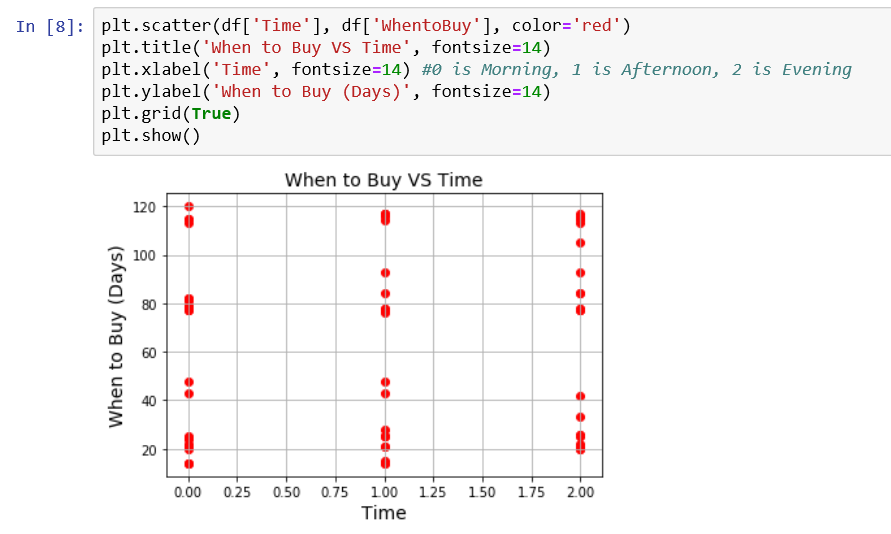
If the files names are not changed, this block of code should run successfully without anymore modifications. Once the code is executed, it should display test.csv as the output. It should output like this:



## Displaying Graphs

The next set of codes will display the graphs of the datasets. These are more for visuals then executing the regression models. You can edit the color, titles, labels, etc.



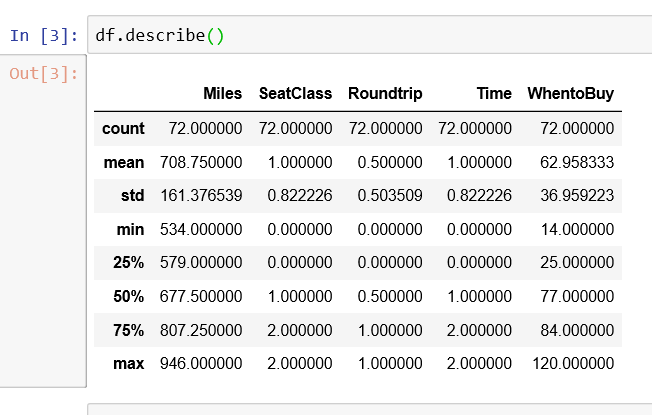


## Analyzing the Datasets

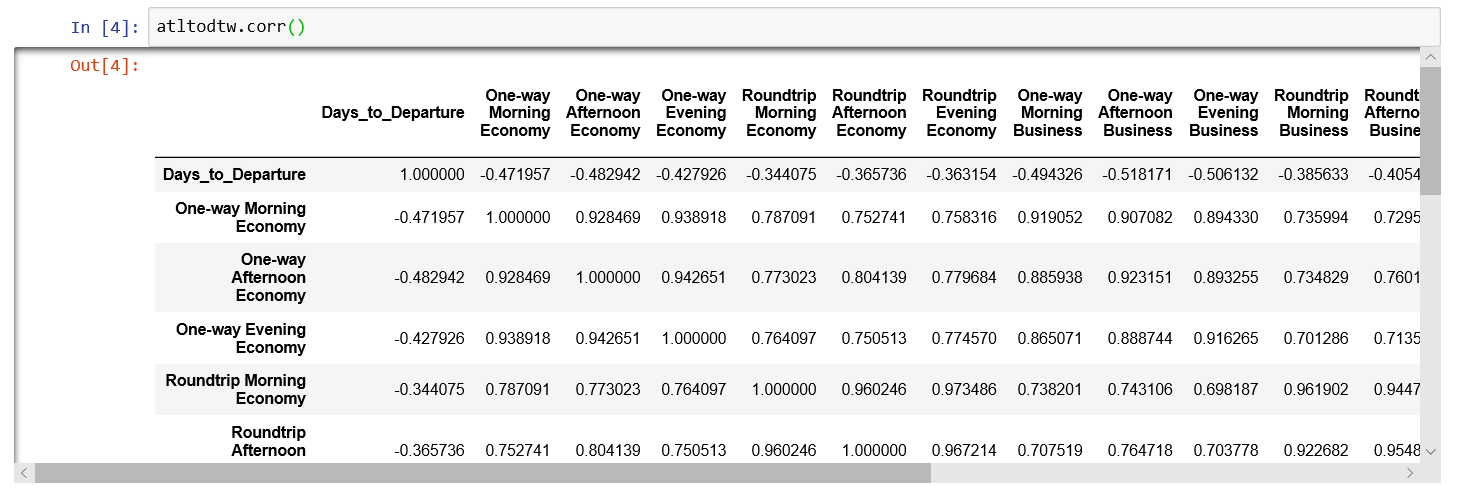
You can analyze the datasets by using the describe and correlation functions.

Use describe() or corr() to analyze the datasets.

For example, you can use the describe function by typing df.describe() which df is the name of the dataset and describe is the function.



In this example, the dataset name is atltodtw and I am using the correlation function:



## Set the Columns for X and Y

Next, we look at setting our X and Y variables. Our goal is to find the Y variable which is the best purchase date. The X axis will be the columns that play an important factor in determining Y.

For the X axis, we will use the columns that contains the parameters that are essential in making a prediction for the best time to buy.

For the Y axis, this will be the column that we are trying to get a prediction on.



## Splitting Data into Training and Test Sets

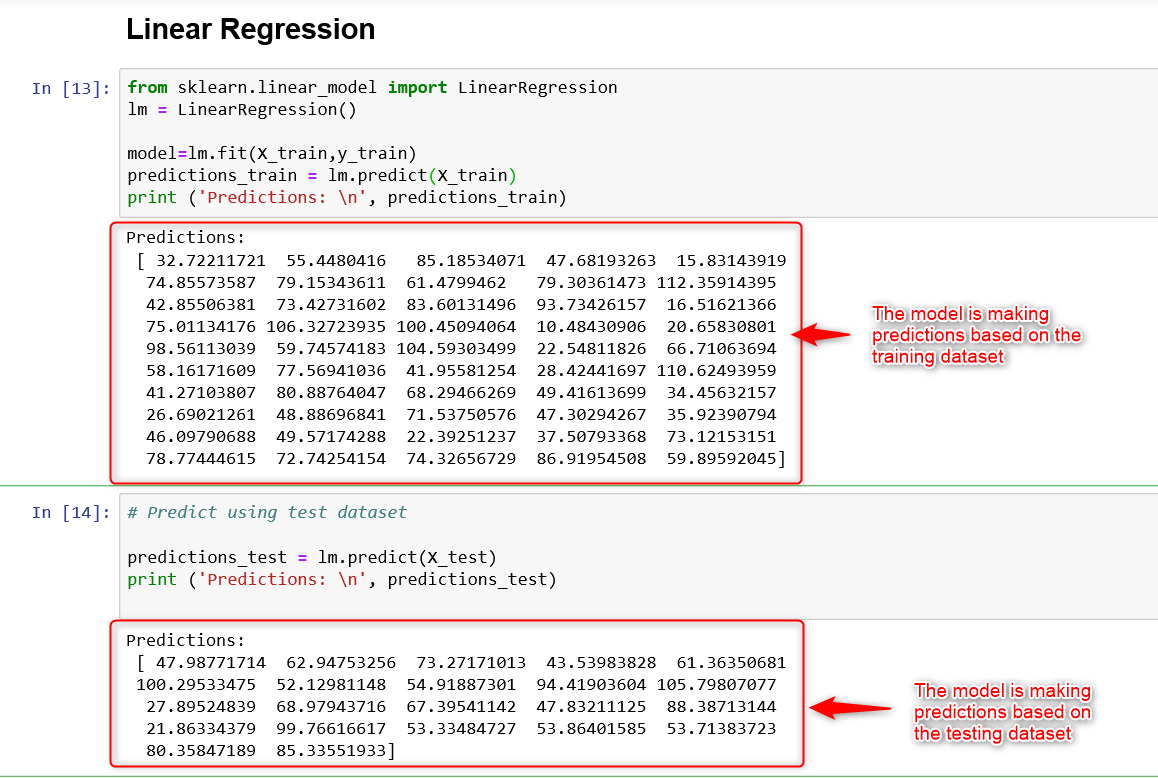
We will use the Sklearn package and code to create our training and testing sets. We will train our models using the training set and have the models make predictions by using the testing set. The recommended split is 70% training/30% testing, however, you are free to change that.



The shape function will display the training/testing dataset size which is saying there are 50 rows and 4 columns for the training set and 22 rows and 3 columns for the testing set.

## Running the Regression Models

Each regression models will be clearly labeled. There will be four blocks of codes for each model. For this example, we will look at the linear regression model. The first two blocks of codes will run the linear regression library from the Sklearn package. This involves fitting the training datasets into the model and allowing the models to make predictions.



After the predictions are made, the regression metrics are executed to determine how well the model did in comparison to the predictions between the training and testing datasets, or in other words, between the actual and predicted values.

The metrics that are evaluated:

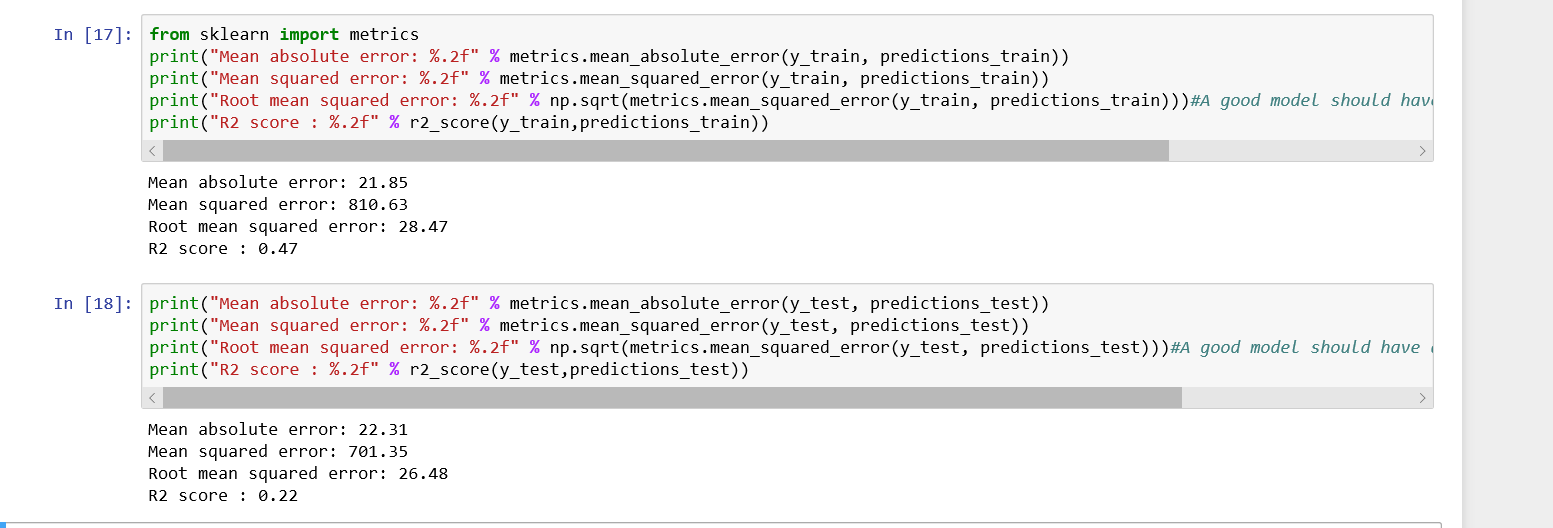
Mean Absolute Error: The sum of the absolute differences between predictions and actual values. It gives an idea of how wrong the predictions were. The measure gives an idea of the magnitude of the error, but no idea of the direction

Mean Squared Error: The Mean Squared Error (or MSE) is much like the mean absolute error in that it provides a gross idea of the magnitude of error.

Root Mean Squared Error: Taking the square root of the mean squared error converts the units back to the original units of the output variable and can be meaningful for description and presentation. This is called the Root Mean Squared Error (or RMSE).

R2 Score: This metric provides an indication of the goodness of fit of a set of predictions to the actual values. In statistical literature, this measure is called the coefficient of determination. This is a value between 0 and 1 for no-fit and perfect fit respectively.

For this project, we decided to use the R2 score (0 to 1) as our prediction accuracy. A high R2 score (1) means a high goodness of fit between the actual and predicted values.



We will use our testing sample to determine the R2 score. Do not use the training set to determine the R2 score as it is notorious for causing in-sample overfitting. For this Linear Regression example, we can see the results of the testing sample:

Mean absolute error: 22.31

Mean squared error: 701.35

Root mean squared error: 26.48

R2 score: 0.22

Continue to evaluate each model.

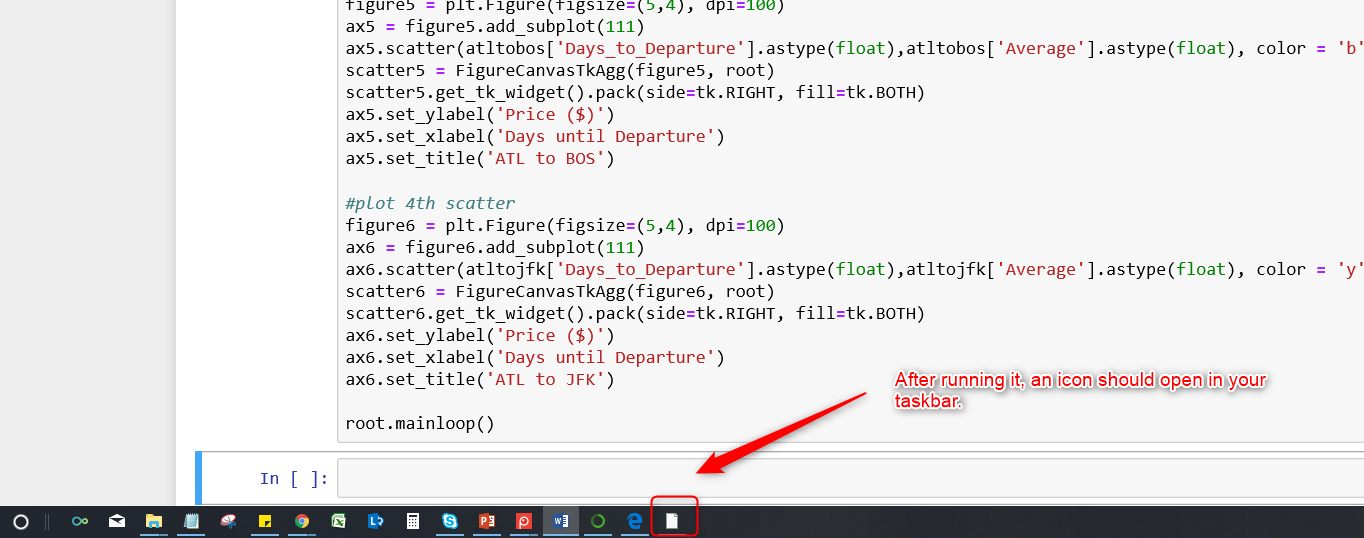
# Predictive Model GUI

## Running the GUI

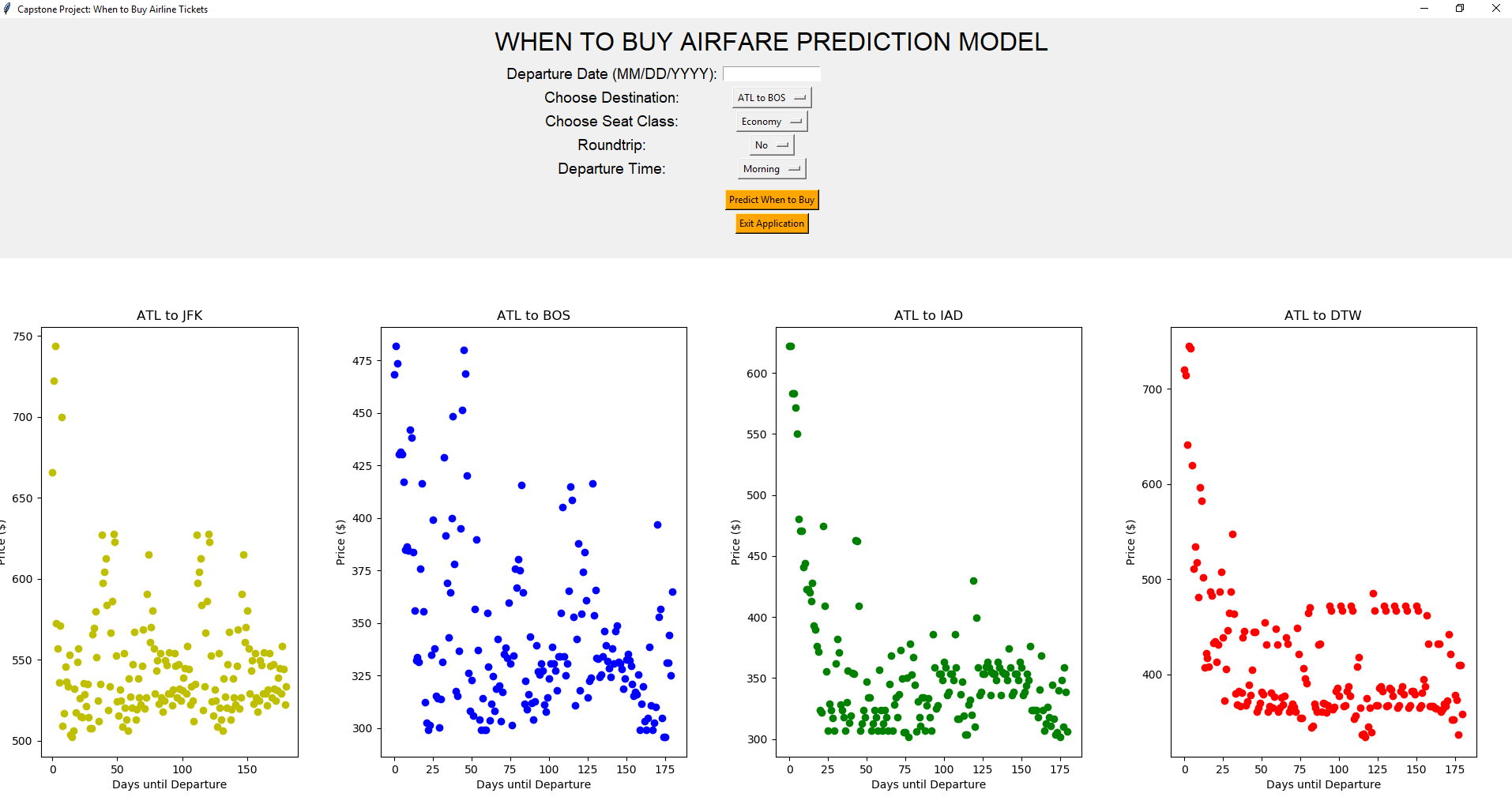
After all the regression models have been run, it is time to choose the model that produced the best prediction accuracy. In this research, we found the Random Forest algorithm to be the best model for our predictive model. The last block of code located in the source code is the Graphical User Interface (GUI) we used to allow a user to input departure date and flight parameters to predict the best date to buy an airline ticket.



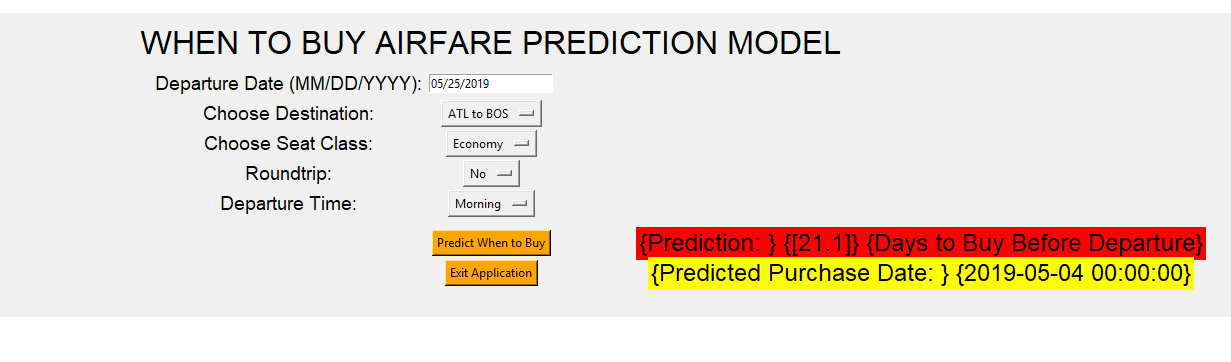
This GUI uses a Python package called tkinter. Click on this block of code and run it. Once it is executed, a GUI window will open.



Opening it will display a GUI that looks similar to this. It will ask the user for a departure data, destination (limited to the four destinations in this research), if it is a roundtrip flight, and departure time.



Once the parameters are selected, the user will click on Predict When to Buy to make a call to the model to make a prediction. The results will be displayed on the GUI.



The results in red are the number of days before the departure day to purchase the airline tickets. The results in yellow is the actual date to purchase the airline ticket.

You will not be able to run any Python code while this GUI is running. You must terminate the application.

Click on Exit Application to terminate the application.

## Modifying Model for the GUI

You can change which model the GUI uses. By default, it is set on the Random Forest model.

In the GUI code, there is a line you can change to change the model that is being used by the GUI. Refer to the screenshot below:



For example, to change to the linear regression model, modify the code to:

lm.predict(Input)

For example, to change to the decisiontree model, modify the code to:

tree\_reg.predict(Input)

Dictionary:

Linear: lm

Decision Tree: tree\_reg

Polynomial: poly

Random Forest: forest\_reg

Support Vector: svrmodel

K-Nearest Neighbor: clf

Logistic: logreg