Internship Schedule:

September 30th to October 2nd

Day to day schedule: 9:00 AM EST – 3:00 PM EST

September 30th: collecting the requirements for the internship and signing the contract

October 1st: Uploading information on the SEVP Portal

October 2nd: Finding a dataset in Kaggle (then clean that data in order to get it ready for analysis)

October 5th to October 9th

Day to day schedule: 9:00 AM EST– 3:00 PM EST

October 5th: Visualizing the data found last Friday. Submitting the data, visualizations for review.

Programming languages used: Python – this programming language was used during my major in several classes – specially in CTBA (Competing Through Business Analytics), Heuristics, and Artificial Intelligence.

The dataset found had columns that did not have enough information, so I had to find another dataset

The name of the dataset is: insurance.csv – it was found on Kaggle

Rows: 1338

Columns: 7

Description of columns:

Age: the age of the insurance holder

Sex: gender of the insurance holder

BMI: body mass index of the insurance holder

Children: the number of children covered by the insurance form

Smoker: binary variable (yes or no)

Region: area of the beneficiary (in the US, northeast, southeast, etc.)

Charges: medical costs billed by the health insurance

Objective of the model:

The purpose of this model is to predict the cost billed by the health insurance company, i.e. the 'charges' column.

The GreatFull Plate wanted me to do something that could benefit me in the long run and in other industries. The let me decide which dataset and which model to implement. At the end of each week, I will submit my work for review to Joseph Evan Markley and Christian Denmark. Also, I will keep communicating with them daily in order to receive guidance and tell them what my plans are.

As mentioned before, the programming language that I'll be using is the Python programming language. However, I also plan on doing it using R in order to compare the two languages. The data visualizations and dashboards will be done using Tableau.

October 6th: 9:00 AM EST – 3:00 PM EST

I finished the basic model – this model does not split the data into test and training data.

The results for this model are:

The R^2 of the model is: 0.7507372027994937

The Mean Squared Error of the model is: 36527659.88568238

The coefficients of the model are:

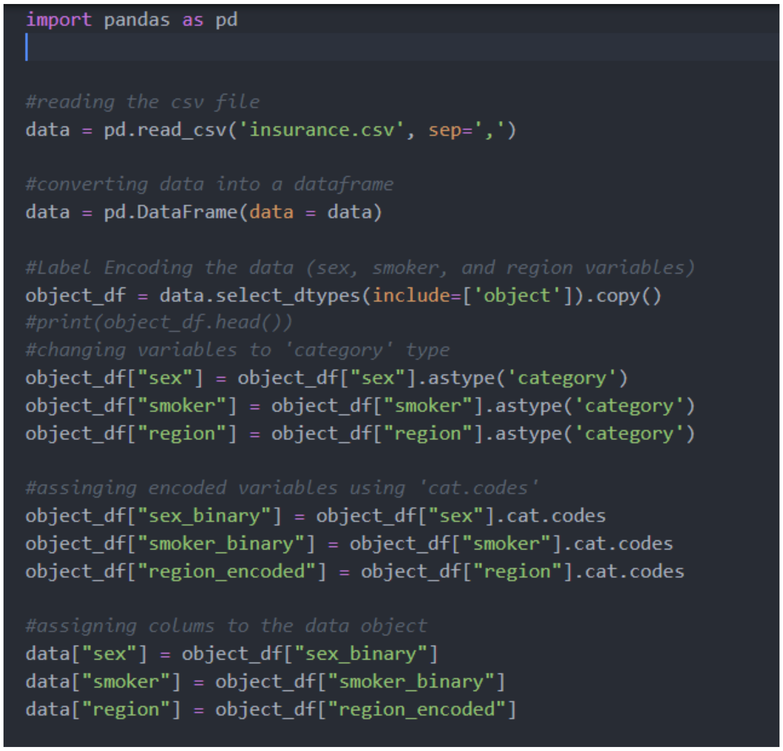
[257.28807486 -131.11057962   332.57013224   479.36939355

23820.43412267 -353.64001656]

From the basic results we can see that the linear regression has a relatively good R squared. However, from the library 'sklearn' I couldn't get the p-values, which are needed in order to determine the significance of the variables.

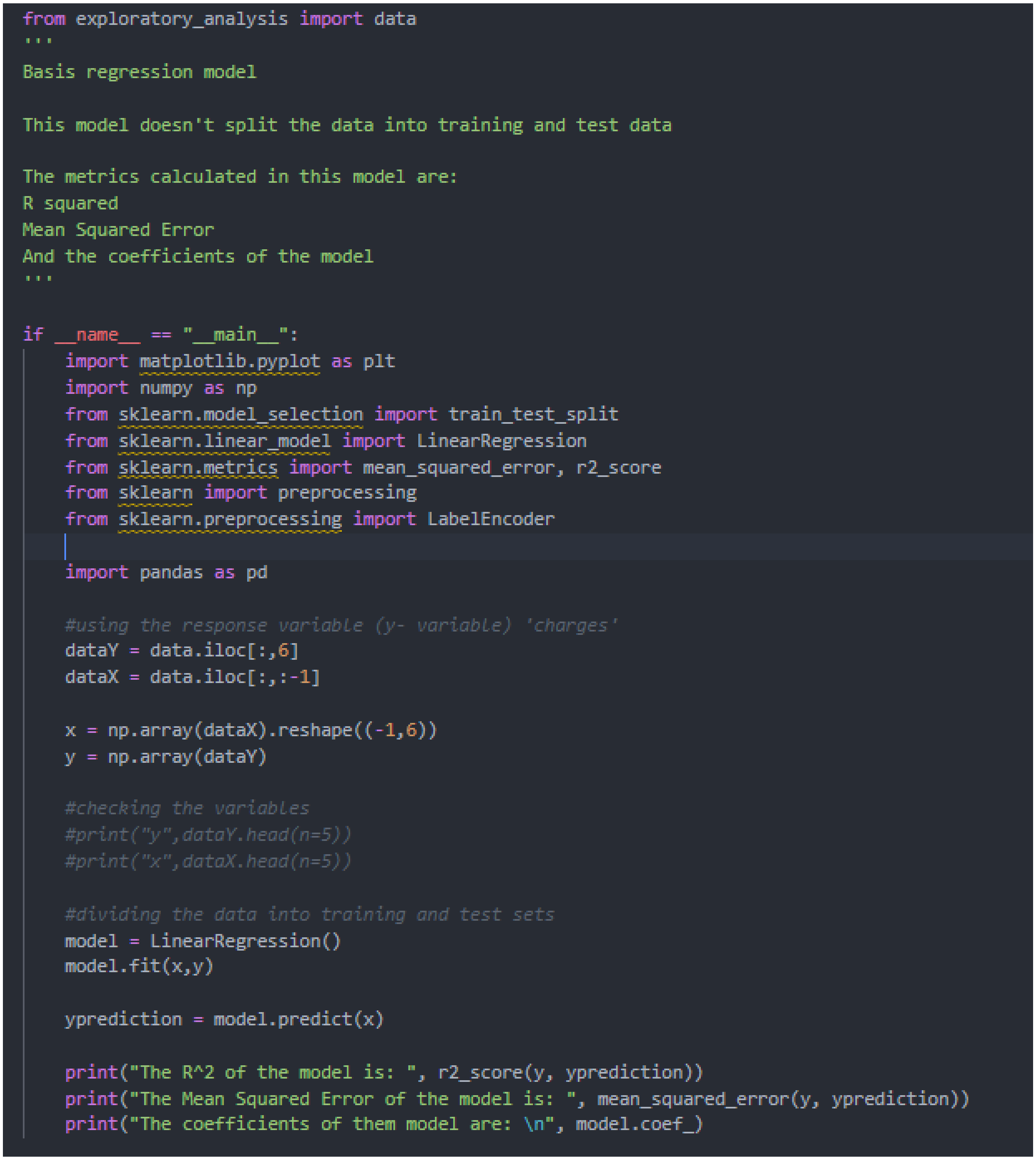
My task today is to find a way to get the p-values in order to have a better model.

I will also try to visualize the data and plot various graphs in order to have a technical and visual approach to conclusions.



```
import pandas as pd


#reading the csv file
data = pd.read_csv('insurance.csv', sep=',')

#converting data into a dataframe
data = pd.DataFrame(data = data)

#Label Encoding the data (sex, smoker, and region variables)
object_df = data.select_dtypes(include=['object']).copy()
#print(object_df.head())
#changing variables to 'category' type
object_df["sex"] = object_df["sex"].astype('category')
object_df["smoker"] = object_df["smoker"].astype('category')
object_df["region"] = object_df["region"].astype('category')

#assinging encoded variables using 'cat.codes'
object_df["sex_binary"] = object_df["sex"].cat.codes
object_df["smoker_binary"] = object_df["smoker"].cat.codes
object_df["region_encoded"] = object_df["region"].cat.codes

#assigning colums to the data object
data["sex"] = object_df["sex_binary"]
data["smoker"] = object_df["smoker_binary"]
data["region"] = object_df["region_encoded"]
```

Data Preprocessing Step – cleaning the data and making it ready for the linear regression.

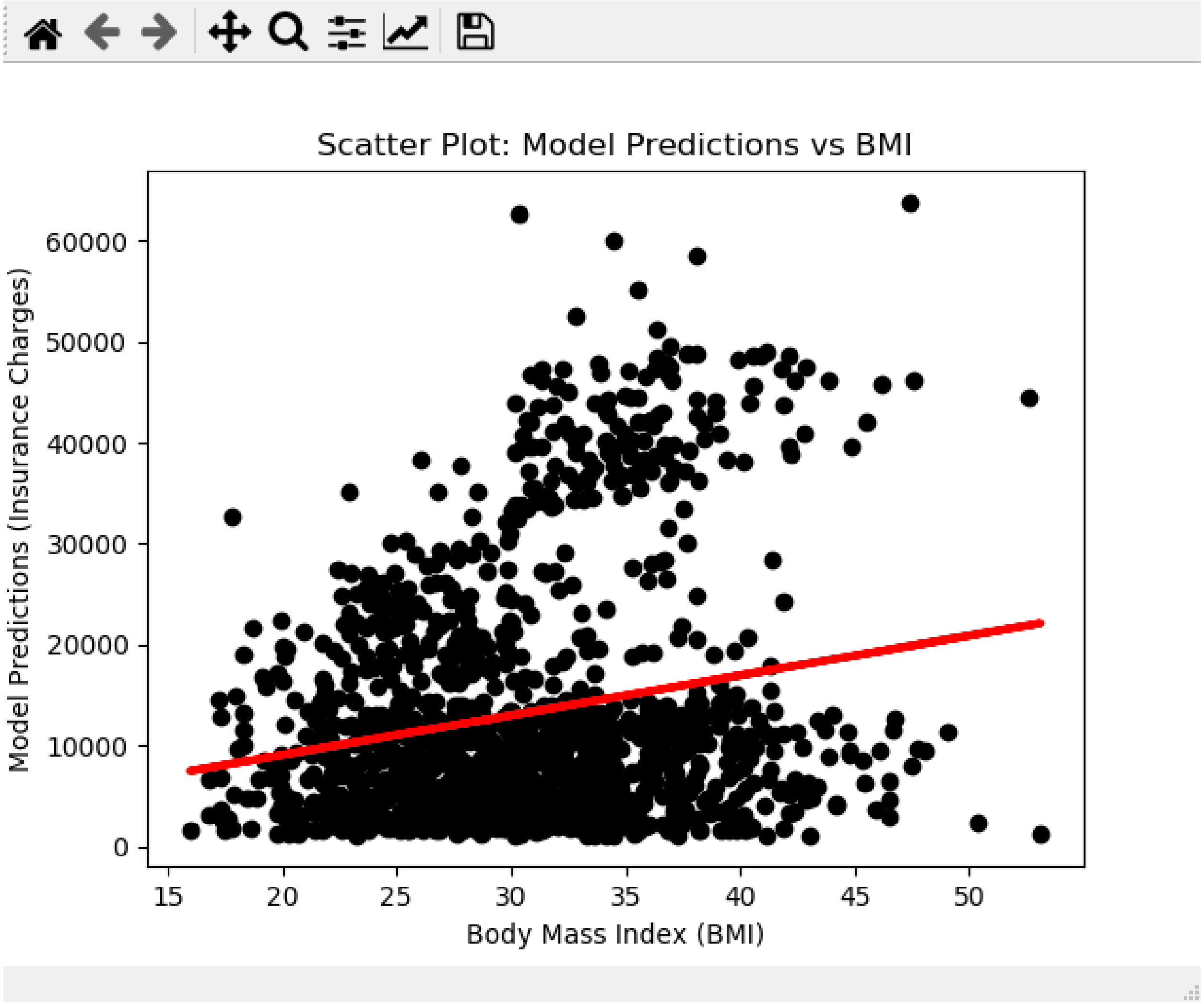


```
from exploratory_analysis import data
'''
Basis regression model

This model doesn't split the data into training and test data

The metrics calculated in this model are:
R squared
Mean Squared Error
And the coefficients of the model
'''

if __name__ == "__main__":
    import matplotlib.pyplot as plt
    import numpy as np
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LinearRegression
    from sklearn.metrics import mean_squared_error, r2_score
    from sklearn import preprocessing
    from sklearn.preprocessing import LabelEncoder

    import pandas as pd

    #using the response variable (y- variable) 'charges'
    dataY = data.iloc[:,6]
    dataX = data.iloc[:,:-1]

    x = np.array(dataX).reshape((-1,6))
    y = np.array(dataY)

    #checking the variables
    #print("y",dataY.head(n=5))
    #print("x",dataX.head(n=5))

    #dividing the data into training and test sets
    model = LinearRegression()
    model.fit(x,y)

    yprediction = model.predict(x)

    print("The R^2 of the model is: ", r2_score(y, yprediction))
    print("The Mean Squared Error of the model is: ", mean_squared_error(y, yprediction))
    print("The coefficients of them model are: \n", model.coef_)
```

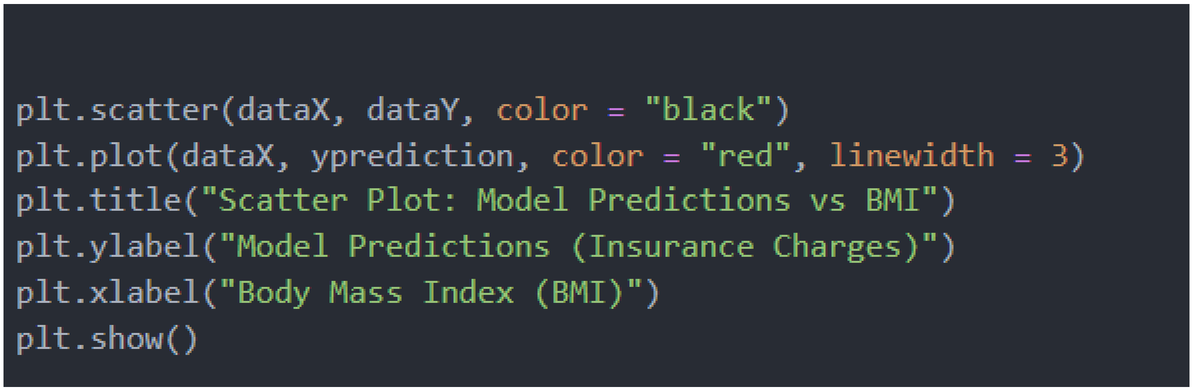
Basic Linear Regression Model. In order to have a more readable model I imported the preprocessing file using the “from exploratory_analysis import data” statement. Then in order for it not to run concurrently with the new file I added the if __name__ == “__main__ statement.

Libraries used: Sklearn, Matplotlib (imported but not used) and Numpy.

Visualization (Body Mass Index – BMI) vs Model Predictions (Charges)



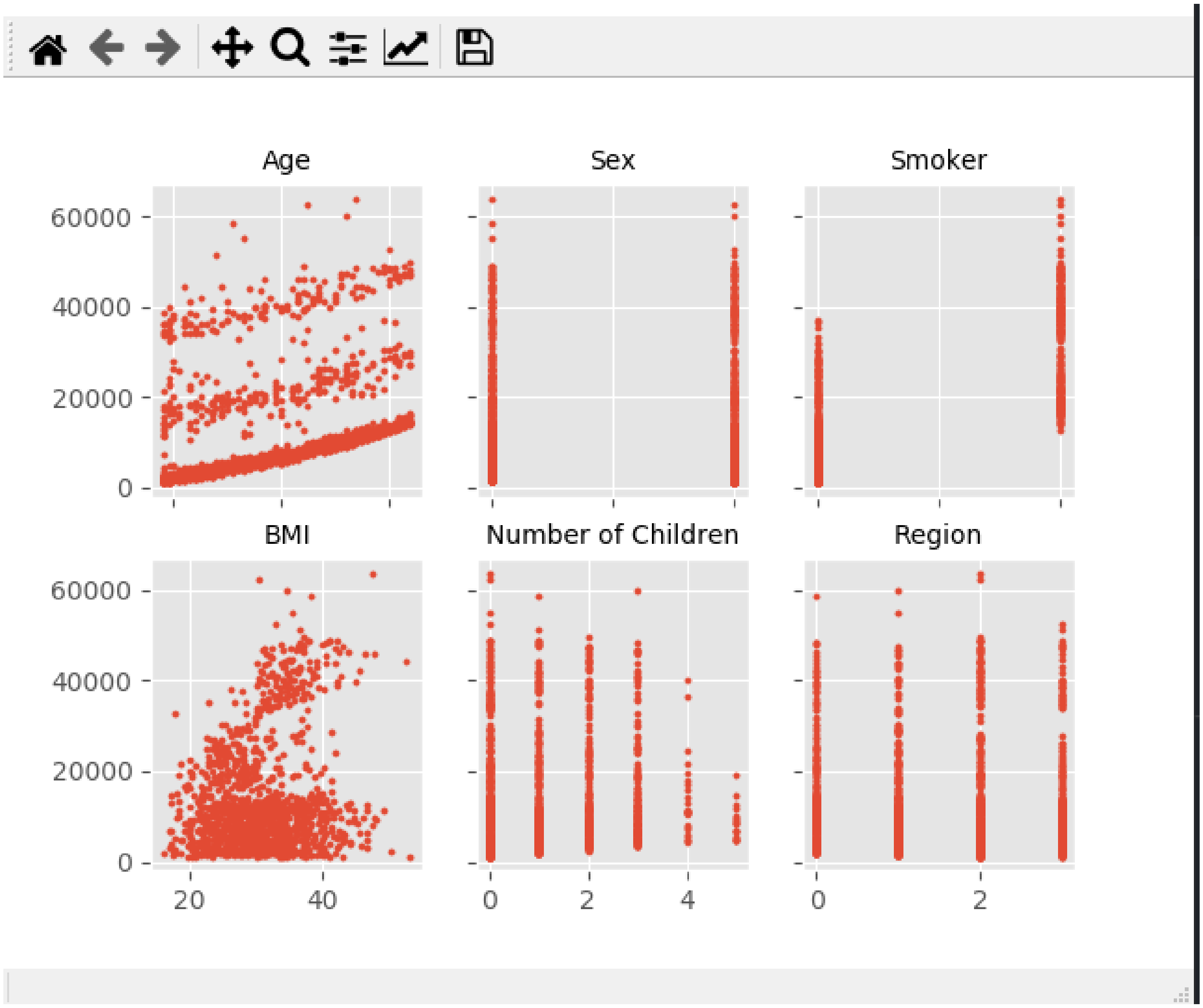
Code to get the above graph:



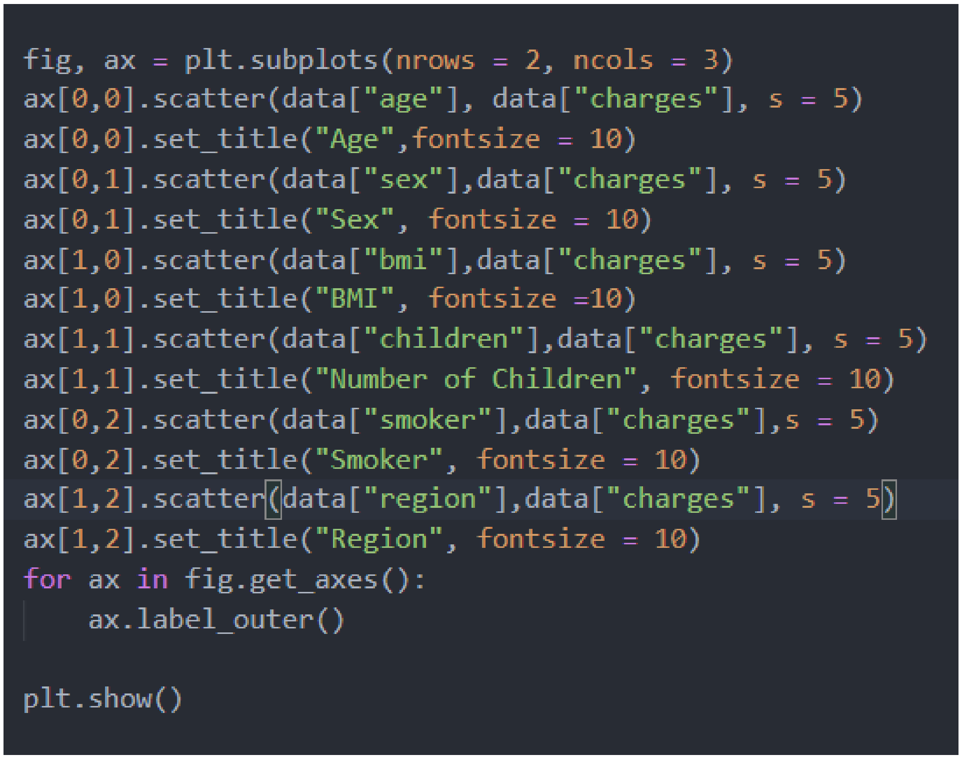
```
plt.scatter(dataX, dataY, color = "black")
plt.plot(dataX, yprediction, color = "red", linewidth = 3)
plt.title("Scatter Plot: Model Predictions vs BMI")
plt.ylabel("Model Predictions (Insurance Charges)")
plt.xlabel("Body Mass Index (BMI)")
plt.show()
```

Where dataX[:,2] column of the dataX DataFrame. And dataY is the response variable.

Scatter Plots:



Code to generate the scatter plots:

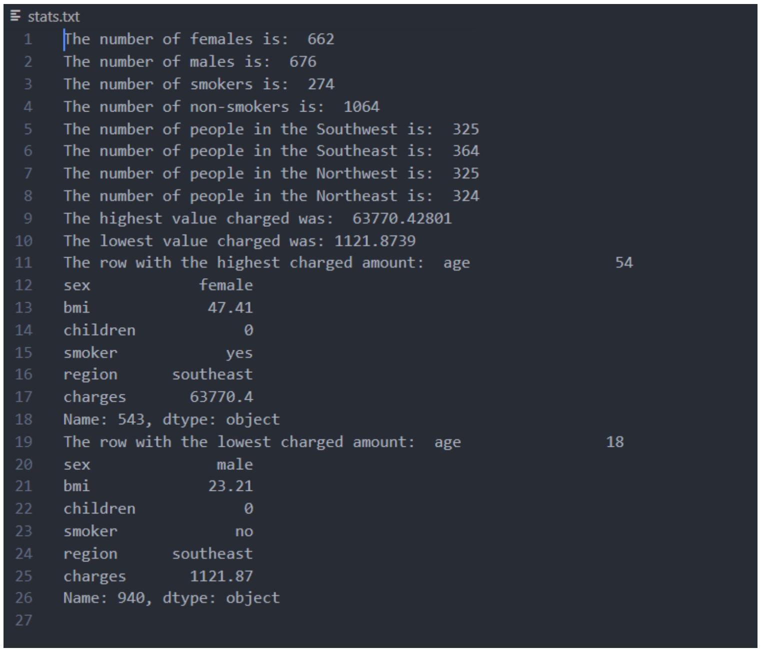


```
fig, ax = plt.subplots(nrows = 2, ncols = 3)
ax[0,0].scatter(data["age"], data["charges"], s = 5)
ax[0,0].set_title("Age",fontsize = 10)
ax[0,1].scatter(data["sex"],data["charges"], s = 5)
ax[0,1].set_title("Sex", fontsize = 10)
ax[1,0].scatter(data["bmi"],data["charges"], s = 5)
ax[1,0].set_title("BMI", fontsize =10)
ax[1,1].scatter(data["children"],data["charges"], s = 5)
ax[1,1].set_title("Number of Children", fontsize = 10)
ax[0,2].scatter(data["smoker"],data["charges"],s = 5)
ax[0,2].set_title("Smoker", fontsize = 10)
ax[1,2].scatter(data["region"],data["charges"], s = 5)
ax[1,2].set_title("Region", fontsize = 10)
for ax in fig.get_axes():
    ax.label_outer()

plt.show()
```

October 7th, 2020 – 9:00 AM EST – 3:00 PM EST: A good question from Christian made me think that since the R^2 squared is high, more than 0.5 and close to 0.8, I should analyze this data further.

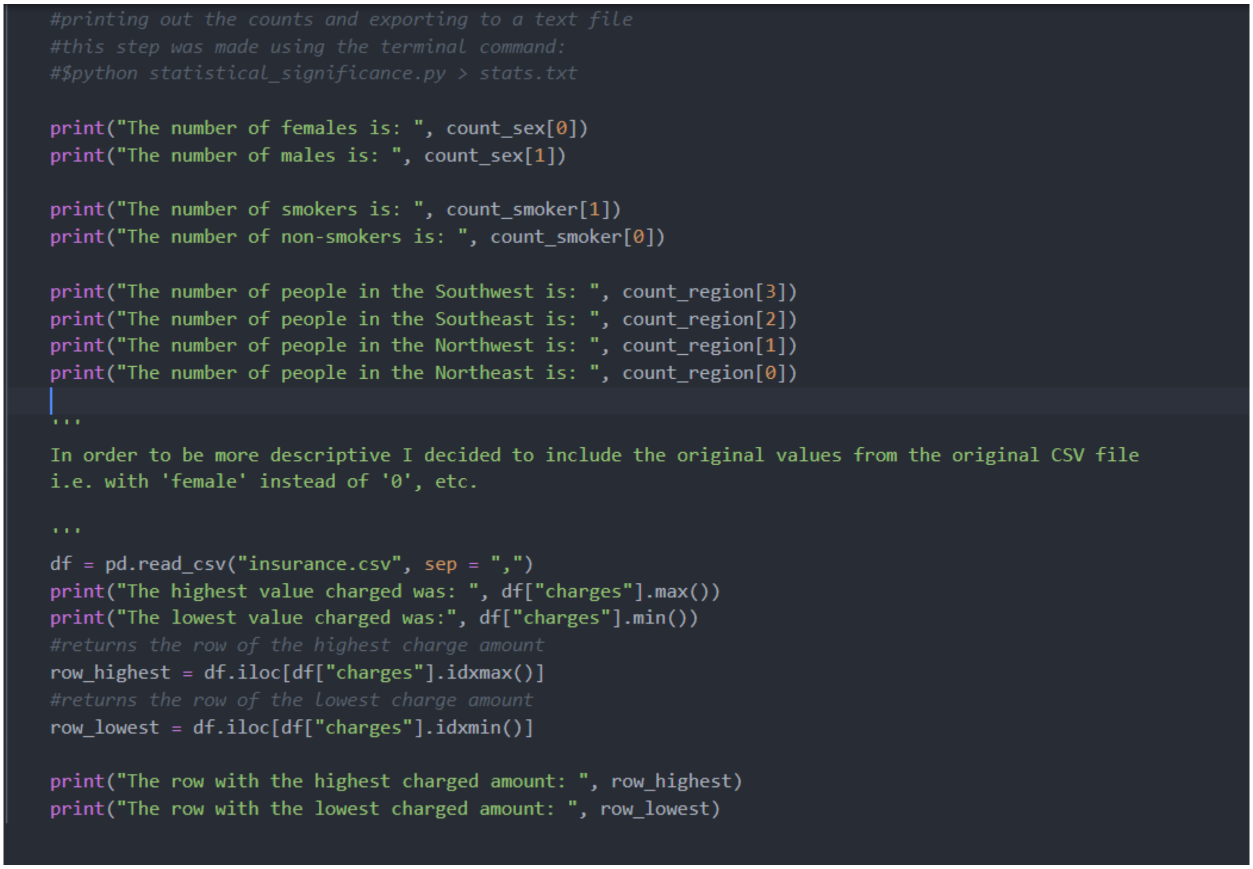
So, I decided to add more plots (histograms) and a text file with important statistics about the data.

The text file:



```
The number of females is:  662
The number of males is:  676
The number of smokers is:  274
The number of non-smokers is:  1064
The number of people in the Southwest is:  325
The number of people in the Southeast is:  364
The number of people in the Northwest is:  325
The number of people in the Northeast is:  324
The highest value charged was:  63770.42801
The lowest value charged was: 1121.8739
The row with the highest charged amount:  age                54
sex               female
bmi                47.41
children               0
smoker               yes
region         southeast
charges          63770.4
Name: 543, dtype: object
The row with the lowest charged amount:  age                18
sex                 male
bmi                23.21
children               0
smoker                no
region         southeast
charges          1121.87
Name: 940, dtype: object

```

Code to generate the text file:



```
#printing out the counts and exporting to a text file
#this step was made using the terminal command:
#$python statistical_significance.py > stats.txt

print("The number of females is: ", count_sex[0])
print("The number of males is: ", count_sex[1])

print("The number of smokers is: ", count_smoker[1])
print("The number of non-smokers is: ", count_smoker[0])

print("The number of people in the Southwest is: ", count_region[3])
print("The number of people in the Southeast is: ", count_region[2])
print("The number of people in the Northwest is: ", count_region[1])
print("The number of people in the Northeast is: ", count_region[0])

'''
In order to be more descriptive I decided to include the original values from the original CSV file
i.e. with 'female' instead of '0', etc.

'''
df = pd.read_csv("insurance.csv", sep = ",")
print("The highest value charged was: ", df["charges"].max())
print("The lowest value charged was:", df["charges"].min())
#returns the row of the highest charge amount
row_highest = df.iloc[df["charges"].idxmax()]
#returns the row of the lowest charge amount
row_lowest = df.iloc[df["charges"].idxmin()]

print("The row with the highest charged amount: ", row_highest)
print("The row with the lowest charged amount: ", row_lowest)
```

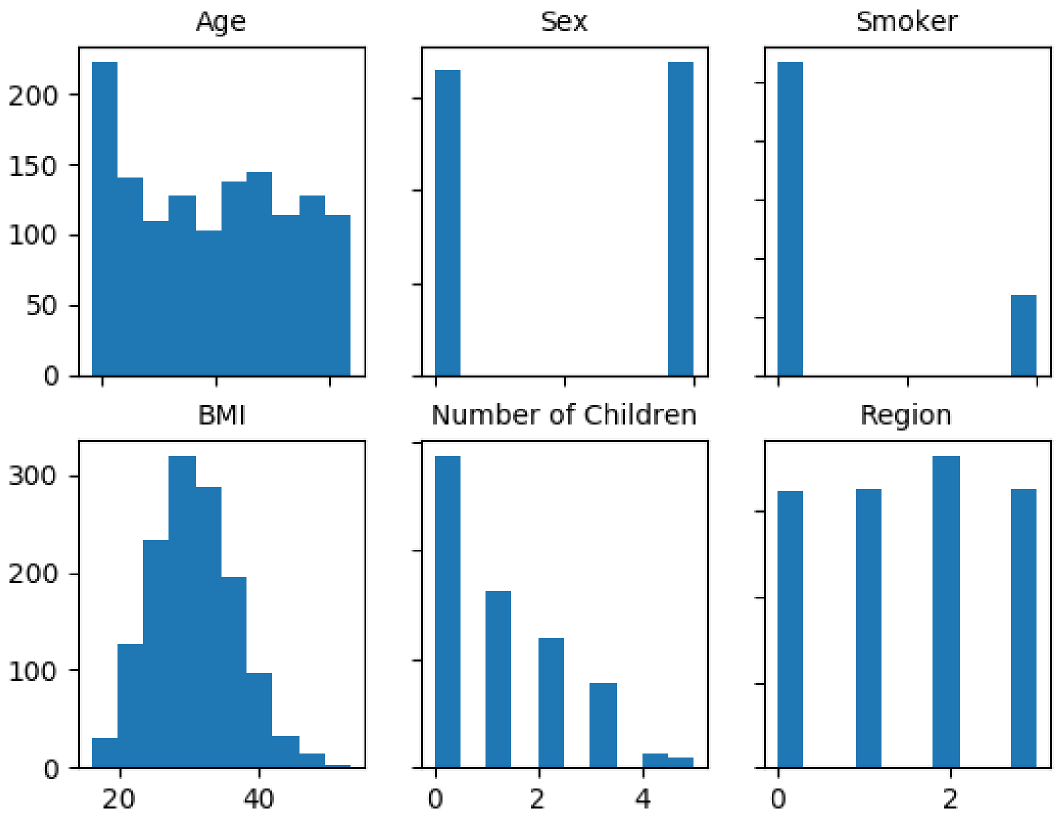
Histograms:

As you can see the histograms are not that descriptive of the problem. In order to have a good model a good understanding of the data is needed. To do this I will continue understanding the data and use other models like PCA analysis.

From the stats.txt file we see that the highest charge was in the Southeast, the person was a female, the age was 54, and she had a high BMI or body mass index.

This information is incomplete as we need to know more about the procedure that was administered to this person. However, this is important because from this information we could infer many correlations in the data. Important to note that ‘correlation does not mean causation’ – this is very important because with delicate data like this we cannot arrive at the wrong conclusion as it could be detrimental for patients and the insurance company.

Like every analysis we must include and not include outliers. If our conclusion are far with/without outliers, then we must reassess the model and ask more question about the data. Outliers are important because they can show that charges could be either too low or too high – even though the bulk of the distribution may be in the middle of the distribution we have to ask the question, ‘why are charges too high’ or ‘why are charges too low’.



Given all this information, I will develop tomorrow the final model (regression, classic regression), random forest, and possibly a classification model. The plan for the next couple of weeks is to continue working on this project until I am ready for presenting my results.

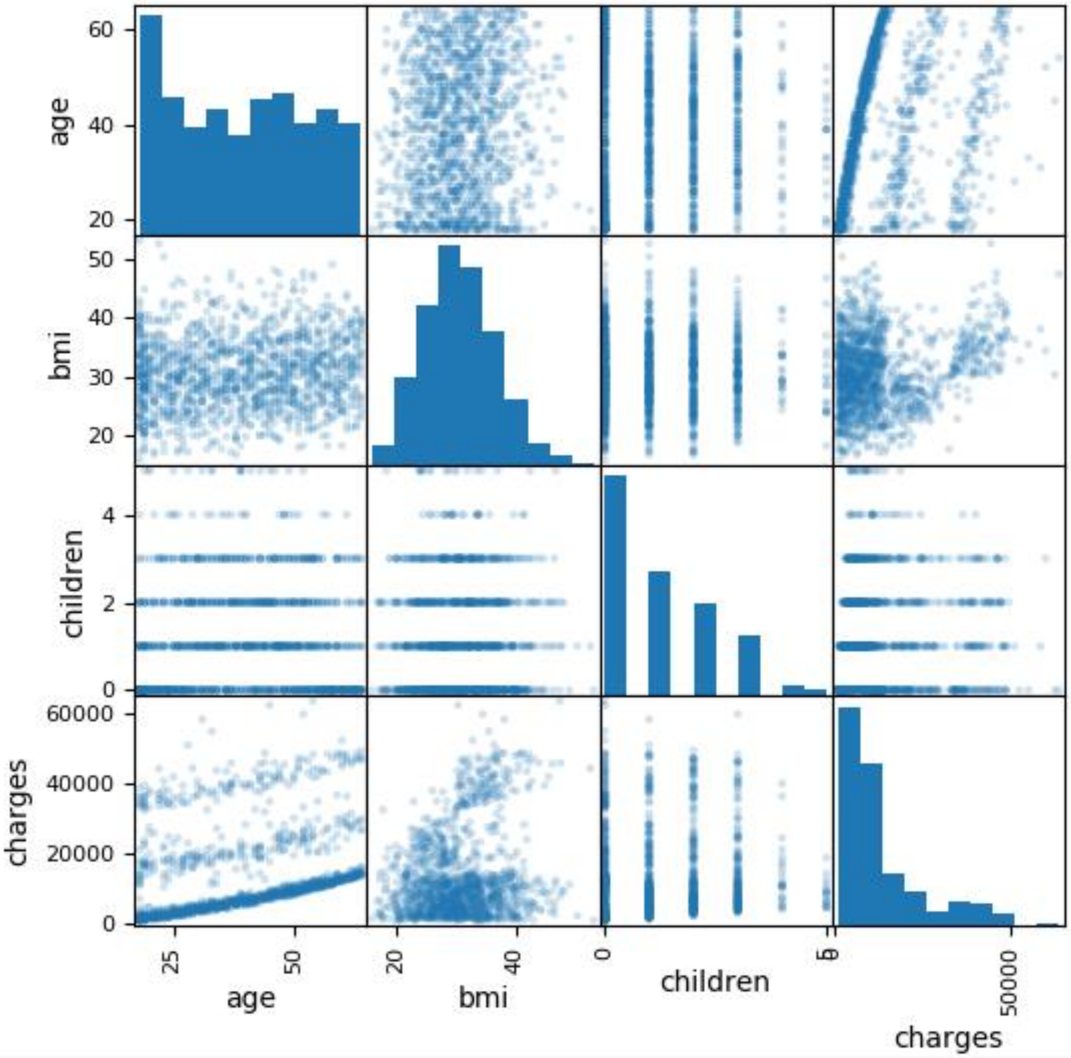
I will also include the code in Python and R in order to compare both programming languages. Finally, if possible, I will include either a Tableau dashboard or a website showcasing the results in an interactive way.

October 8th, 2020– 9:00 AM EST – 3:00 PM EST

As mentioned yesterday, today I developed the final regression model – however I did not complete the random forest.

The final regression model is the first model that I developed, with the R^2 squared ~ 0.75 was done using only the data given (no shuffling of the data and no K-Fold Cross Validation). This is the final 'basic' model, however in order to compare both I will add the K-Fold Cross Validation.

The plots that I developed today are:



This is a scatter plot done with the pandas library in Python.

The diagonal is a histogram of the distribution of the variable (against itself). This scatter plot matrix is important because we can see if the variables are correlated or not. However, it also has downsides. For example, it is not easy to read in one color and the information gets lost sometimes.

Therefore, different plots that convey the same message are important. The plots that I included before bare some resemblance to this last plot – they are the same but not included in one picture. This is the benefit of having the scatter plot matrix, that instead of having multiple plots (and developing that code in several lines, debugging it, etc.) it is better to include all of them in one single plot.

Including several plots is a good idea, especially if they relate to each other.

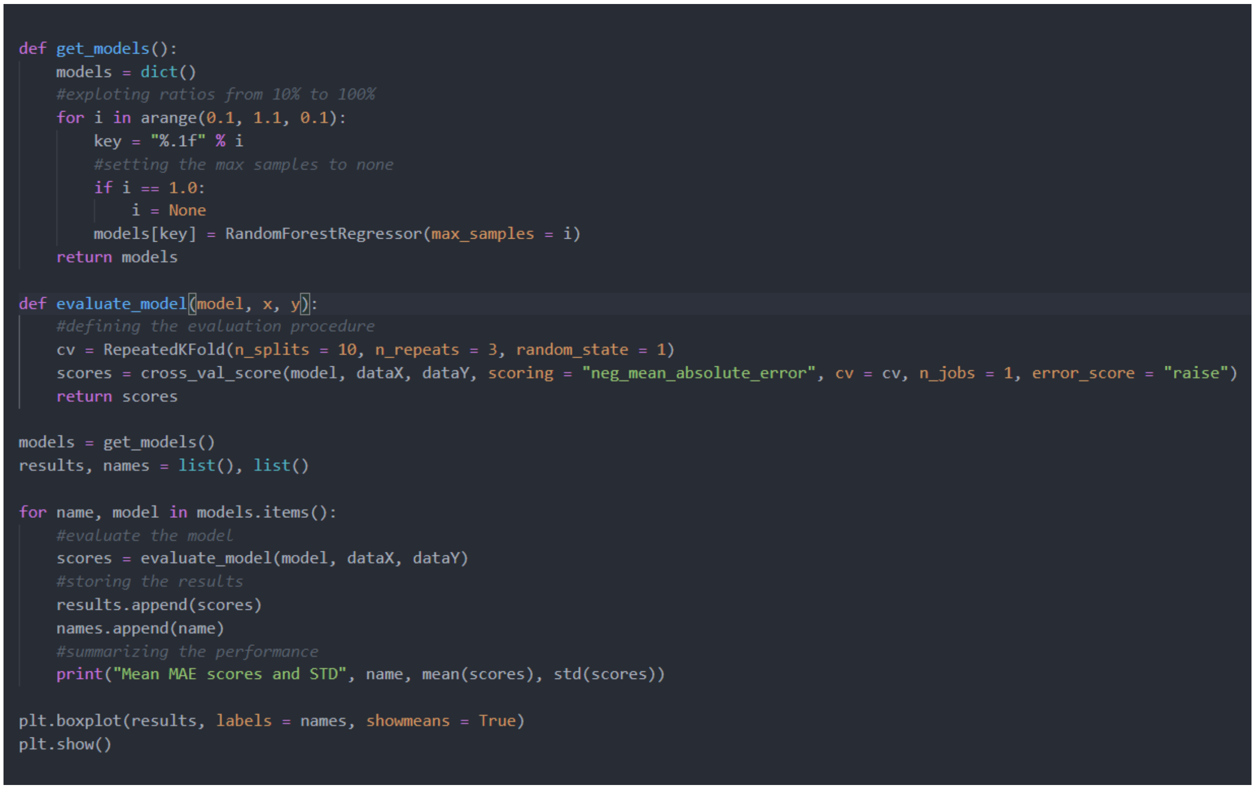
*Important to note that the regression that I concluded earlier this week was against all predictors. Using K-Fold, p-values, and one predictor at a time (or a combination) will be better to arrive at the right conclusion.

The plans for tomorrow include basic random forest model, and individual regressions with different combinations.

October 9th, 2020 – 9:00 AM EST – 3:00 PM EST

Today I completed the random forest model. I took me a while to complete as I was having problems with the plots and the model itself. However, I completed it thanks to code that I adapted from the website, Machine Learning Mastery.

URL: https://machinelearningmastery.com/random-forest-ensemble-in-python/



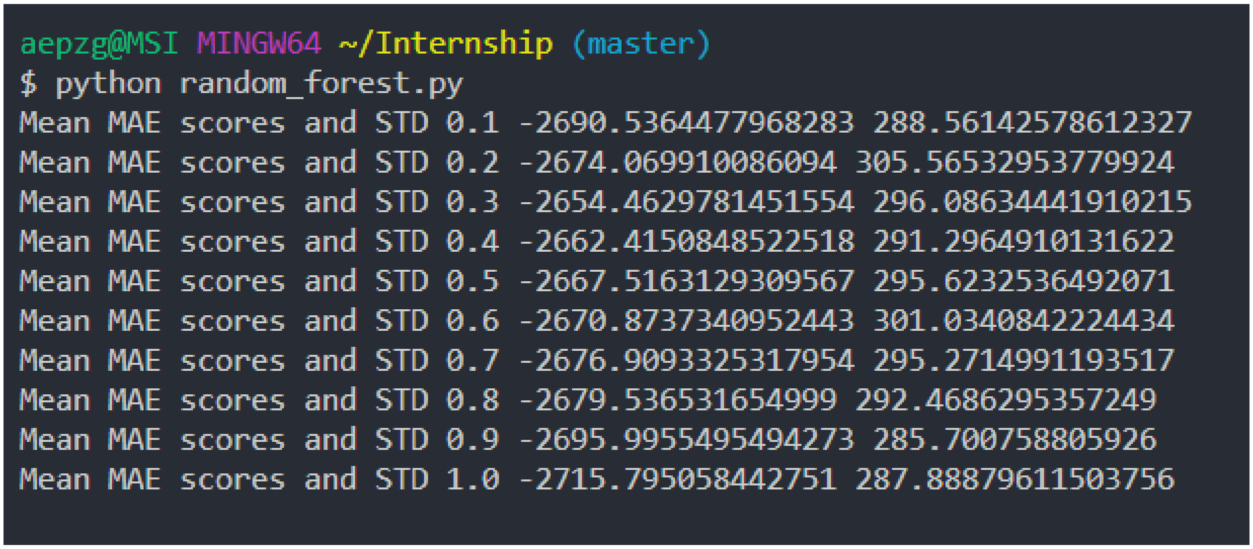
```
def get_models():
    models = dict()
    #exploting ratios from 10% to 100%
    for i in arange(0.1, 1.1, 0.1):
        key = "%.1f" % i
        #setting the max samples to none
        if i == 1.0:
            i = None
        models[key] = RandomForestRegressor(max_samples = i)
    return models

def evaluate_model(model, x, y):
    #defining the evaluation procedure
    cv = RepeatedKFold(n_splits = 10, n_repeats = 3, random_state = 1)
    scores = cross_val_score(model, dataX, dataY, scoring = "neg_mean_absolute_error", cv = cv, n_jobs = 1, error_score = "raise")
    return scores

models = get_models()
results, names = list(), list()

for name, model in models.items():
    #evaluate the model
    scores = evaluate_model(model, dataX, dataY)
    #storing the results
    results.append(scores)
    names.append(name)
    #summarizing the performance
    print("Mean MAE scores and STD", name, mean(scores), std(scores))

plt.boxplot(results, labels = names, showmeans = True)
plt.show()
```

This is the code that resulted in the random forest model. The model on the website was a random forest classifier model, so I had to change that to a regressor model using the MAE (mean absolute error) as a scoring mechanism and the “RandomForestRegressor” command.

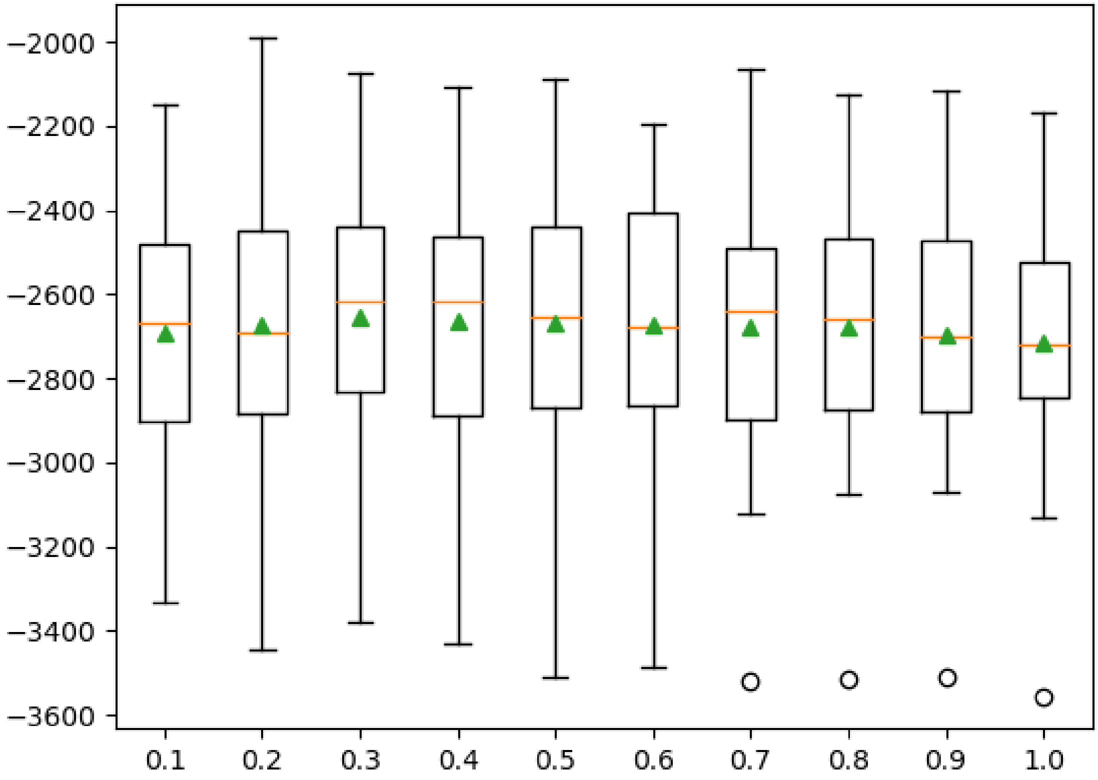
The for loop in the get_models() function returns several models from 10% to 100% in 10% increments. The evalutate_model() function takes three parameters, evaluates the data using cross validation, and returns scores using the mean absolute value.

Finally, a for loop iterates through the models’ dictionary to assign the corresponding score and print out the MAE (mean and standard deviation).

The result is the following plot.



```
aepzg@MSI MINGW64 ~/Internship (master)
$ python random_forest.py
Mean MAE scores and STD 0.1 -2690.5364477968283 288.56142578612327
Mean MAE scores and STD 0.2 -2674.069910086094 305.56532953779924
Mean MAE scores and STD 0.3 -2654.4629781451554 296.08634441910215
Mean MAE scores and STD 0.4 -2662.4150848522518 291.2964910131622
Mean MAE scores and STD 0.5 -2667.5163129309567 295.6232536492071
Mean MAE scores and STD 0.6 -2670.8737340952443 301.0340842224434
Mean MAE scores and STD 0.7 -2676.9093325317954 295.2714991193517
Mean MAE scores and STD 0.8 -2679.536531654999 292.4686295357249
Mean MAE scores and STD 0.9 -2695.9955495494273 285.700758805926
Mean MAE scores and STD 1.0 -2715.795058442751 287.88879611503756
```



*10 models with their MAEs and the corresponding box plot from the model.

From the box plots we see that in the 0.7, 0.8, 0.9, and 1.0 models there are outliers, which are the dots just above the x-axis. The box plot also visualizes the distribution of the MAE scores – which correlate to the values shown in the picture above the box plots.

My conclusion is that the mean average error of this model is around 2600. This roughly means that on average we will be wrong in our predictions by $2,600. Since this data deals with insurance claims, it is not a good estimate as the person receiving treatment, or the insurance company will be forced to pay ±$2,600.

I did not start the individual regressions, but I hope to finish the analysis by Friday next week. The next step are the individual regressions, more plots, and then the Tableau Dashboard.

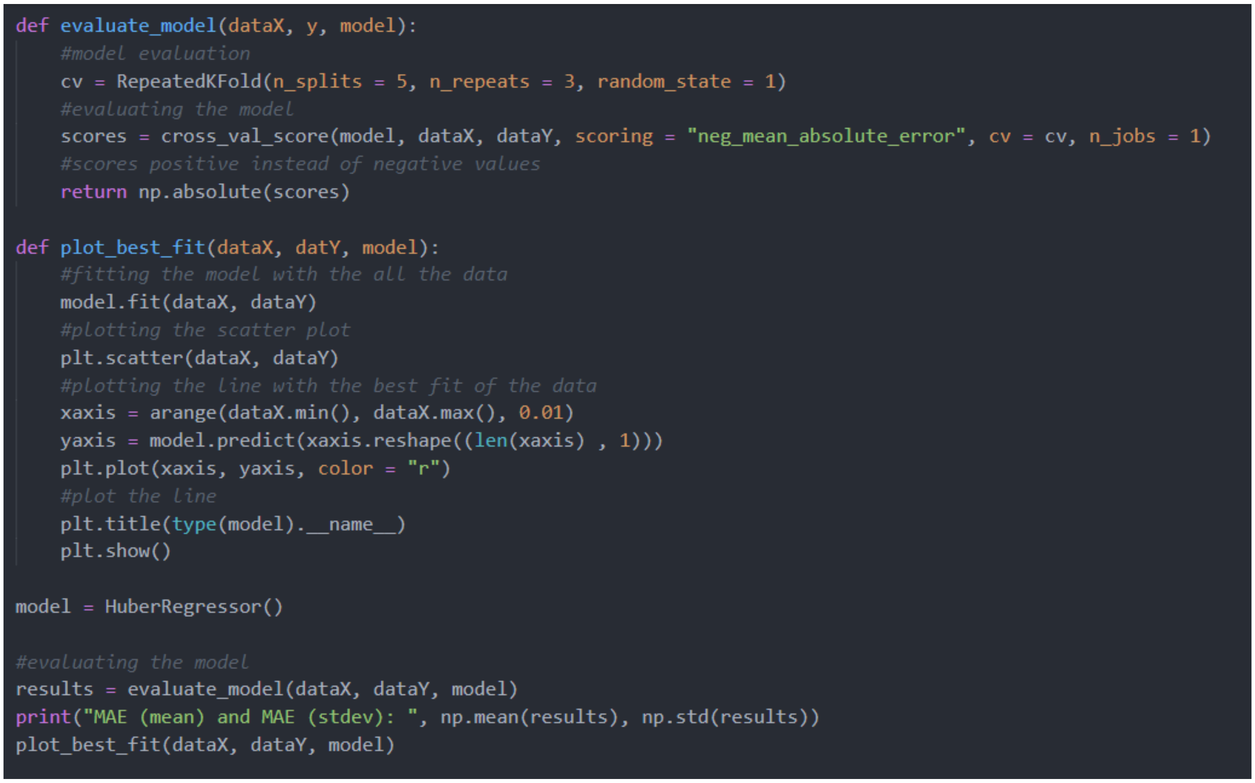
October 12, 2020 – 9:00 AM EST – 3:00 PM EST

Libraries used: SKlearn, Numpy, Pandas, and Matplotlib (for plotting graphs)

Works Cited: Machine Learning Mastery website by Jason Brownlee.

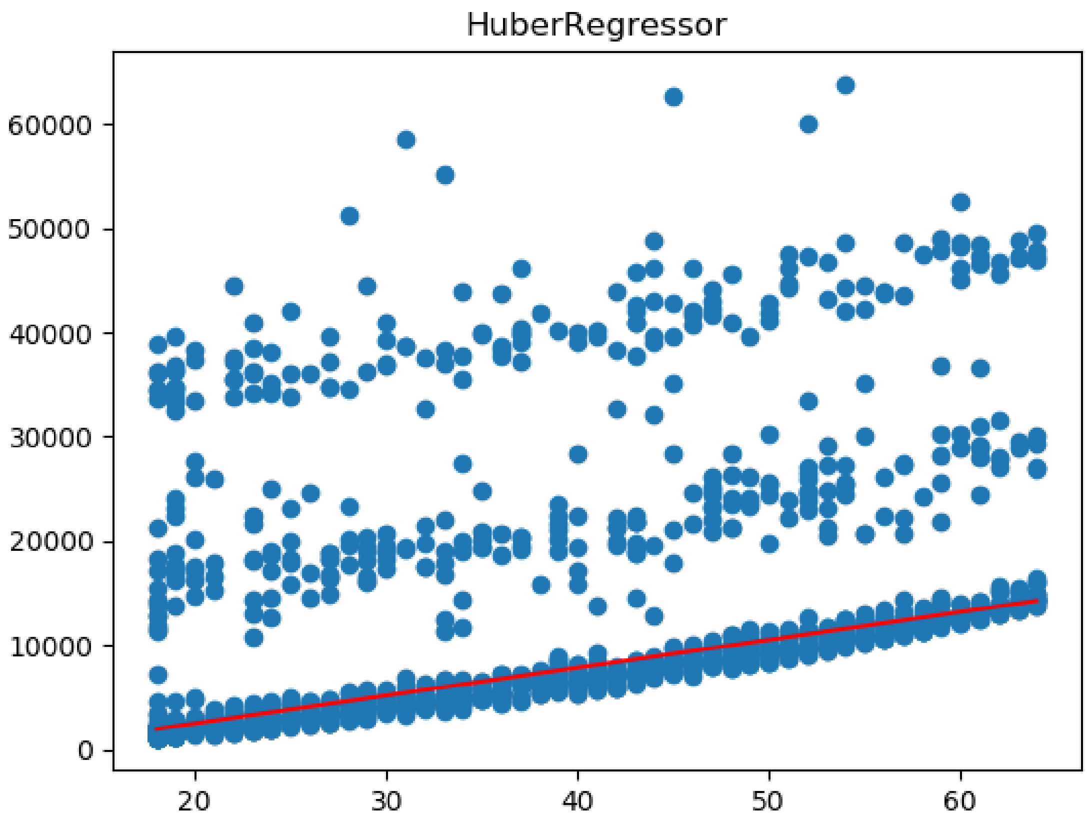
URL: https://machinelearningmastery.com/robust-regression-for-machine-learning-in-python/#:~:text=Regression%20is%20a%20modeling%20task,most%20successful%20being%20linear%20regression.

Code Screenshot:



```
def evaluate_model(dataX, y, model):
    #model evaluation
    cv = RepeatedKFold(n_splits = 5, n_repeats = 3, random_state = 1)
    #evaluating the model
    scores = cross_val_score(model, dataX, dataY, scoring = "neg_mean_absolute_error", cv = cv, n_jobs = 1)
    #scores positive instead of negative values
    return np.absolute(scores)

def plot_best_fit(dataX, datY, model):
    #fitting the model with the all the data
    model.fit(dataX, dataY)
    #plotting the scatter plot
    plt.scatter(dataX, dataY)
    #plotting the line with the best fit of the data
    xaxis = arange(dataX.min(), dataX.max(), 0.01)
    yaxis = model.predict(xaxis.reshape((len(xaxis) , 1)))
    plt.plot(xaxis, yaxis, color = "r")
    #plot the line
    plt.title(type(model).__name__)
    plt.show()

model = HuberRegressor()

#evaluating the model
results = evaluate_model(dataX, dataY, model)
print("MAE (mean) and MAE (stdev): ", np.mean(results), np.std(results))
plot_best_fit(dataX, dataY, model)
```

Individual Regression 1: Age vs. Charges (using the Huber Regressor technique)



What is the Huber Regression technique?

Huber Regression is a technique that assigns less weight to outliers.

Model Results:

MAE (mean squared error, mean): 6,738

MAE (standard deviation from the mean): 553.04

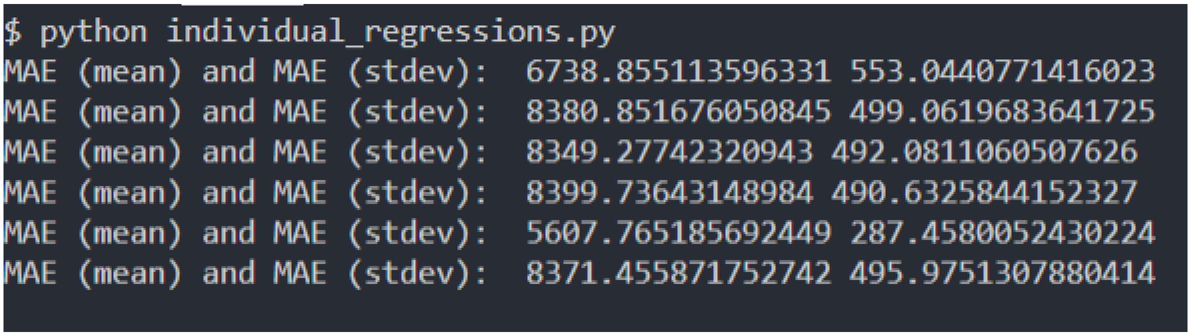
Conclusions:

This was only the result of one variable (age), however, the plot indicates the commonsense approach that insurance charges increase as the person ages. It is important to always plot graphs even if we know that such conclusion is correct – we always want to maintain data integrity and prove our conclusions with facts not with our assumptions.

In order to achieve this Huber Regressor, I had to reshape the data with the “dataX.reshape((-1,1))” command. This is because the model is expecting a 2D array – and dataX is data frame, thus it must be converted to a numpy array and then reshaped into a 1D array. At first, I didn't know why I kept getting an error, but then I realized that the data type conversions had to be done.

During this week I will continue working on the regressions, visualizations, and compare the linear regression predictions vs. random forest predictions. Finally, a Python vs. R comparison and a dashboard using Tableau.

October 14th, 2020 – 9:00 AM EST– 3:00 PM EST

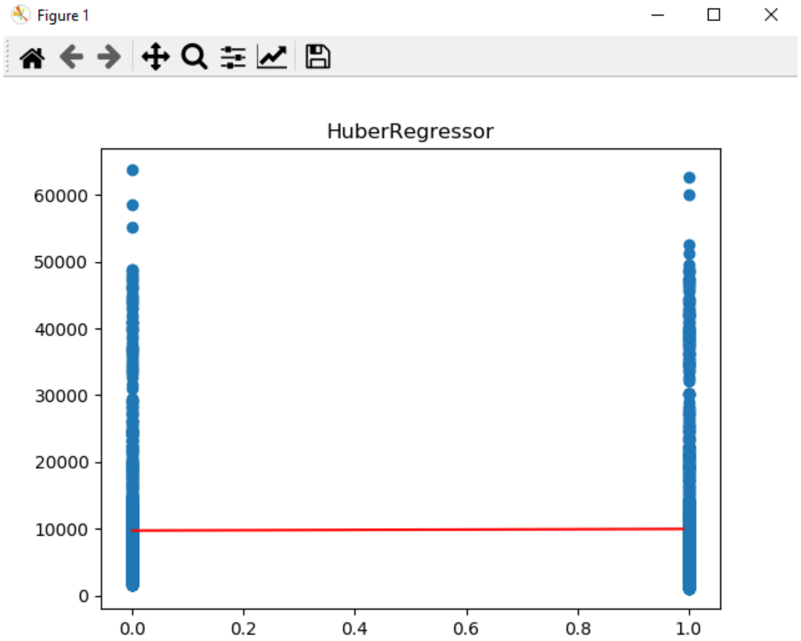


```
$ python individual_regressions.py
MAE (mean) and MAE (stdev):  6738.855113596331 553.0440771416023
MAE (mean) and MAE (stdev):  8380.851676050845 499.0619683641725
MAE (mean) and MAE (stdev):  8349.27742320943 492.0811060507626
MAE (mean) and MAE (stdev):  8399.73643148984 490.6325844152327
MAE (mean) and MAE (stdev):  5607.765185692449 287.4580052430224
MAE (mean) and MAE (stdev):  8371.455871752742 495.9751307880414
```

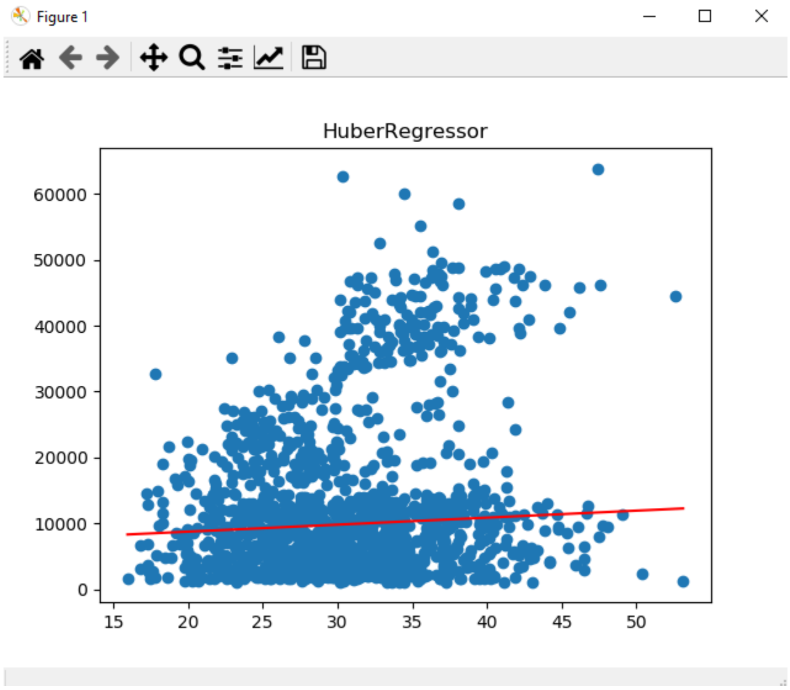
The above picture is in order and corresponds to the predictors in the data – age, sex, body mass index, children, smoker, and region.

The lowest MAE (mean) and MAE (stdev) was for the region variable– there are four regions in the data (southwest, southeast, northwest, and northeast).

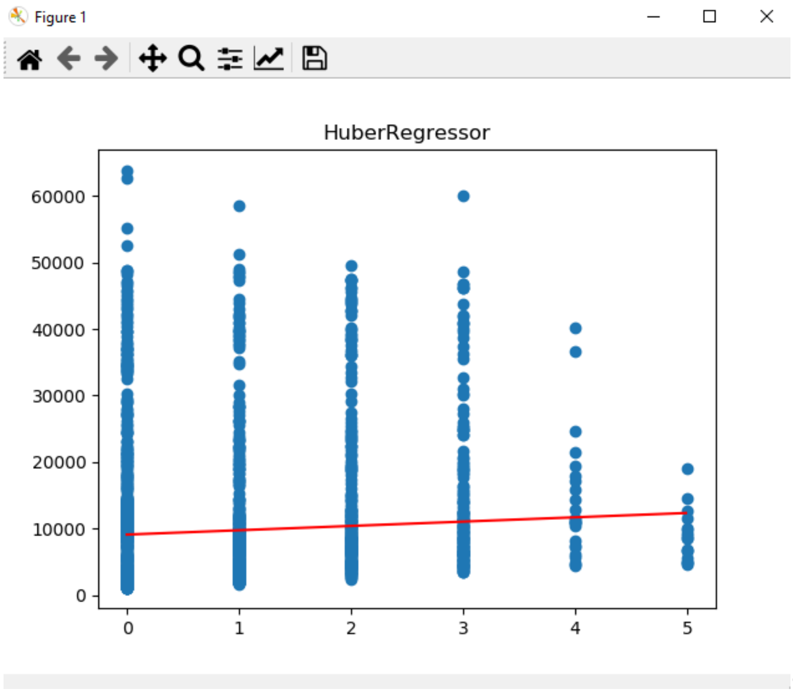
The plots from the individual regressions are (excluding the age variable as that was done yesterday):



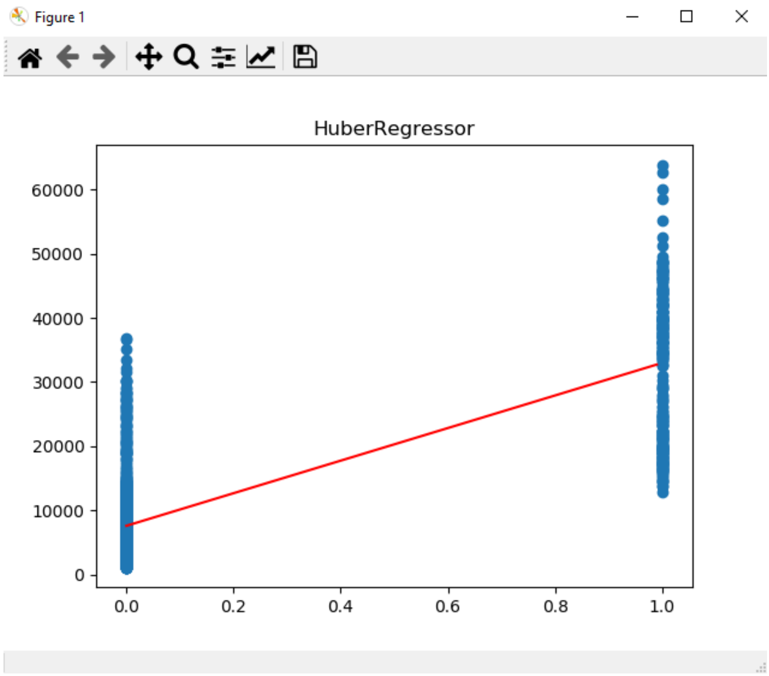
sex



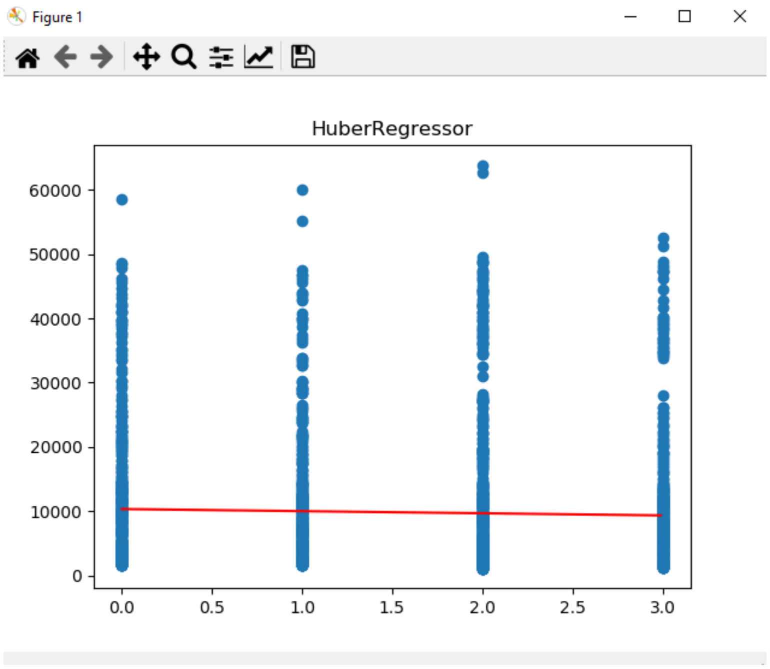
Body Mass Index



Children



Smoker



Region

From these graphs we can conclude that smokers' (1) insurance charges are greater than those who don't smoke. Even though this is an obvious conclusion, it is important to do the regression and plot the charts. This way we could also infer that the other regressions are correct, and we could continue the analysis.

Another conclusion is that most of the 'expensive' charges occur for people whose body index is around 25 – 45. According to the Centers for Disease Control and Prevention (CDC), the body mass index classification is as follows:

BMI < 18.5 is the underweight range

BMI from 18.5 to <25 is the normal range

BMI from 25 to <30 is the overweight range

BMI from 30 or HIGHER is the obese range

Source url: https://www.cdc.gov/obesity/adult/defining.html

With this clarification it makes sense that since obesity is the precursor of many diseases, the range from 25 to 45 is going to be charged more insurance charges.

The number of children is not a good variable for the model, however more analysis is needed to drop it from further analysis.

The sex, male or female, is almost identical – however, I will separate male and female and provide more insight as to the amount overweight, how many smokers in each gender, and the age.

Tomorrow I will continue the analysis and provide more insight in my findings. I will also document/comment my programs, add a data dictionary, provide a list of the definitions of some of the methods and models used in this report, and finally provide a list of all the libraries used in the programs.

October 15th, 2020 – 9:00 AM EST – 3:00 PM EST

Today I documented the several programs that I had written so far. I included as many comments as possible as well as information in how to run certain programs – some were run from the terminal and others were run using an IDE (integrated development environment).

Data dictionary:

Age: the age of the insurance holder

Sex: gender of the insurance holder (male or female) – female = 0, male = 1

BMI: body mass index of the insurance holder

Children: the number of children covered by the insurance form

Smoker: binary variable (yes or no) – smoker = 1, non-smoker = 0

Region: area of the beneficiary (in the US, northeast, southeast, etc.)

- Southwest 3
- Southeast 2
- Northwest 1
- Northeast 0

Charges: medical costs billed by the health insurance

File names and their description:

exploratory_analysis.py: this is the main file that reads the CSV file and explores the data. In this file the sex, smoker, and region columns are converted to numerical type in order to get it ready for analysis.

regression.py: first regression model. This model does not split the data into training or test data. Therefore, this model could be overfitting the data – i.e. giving it a high accuracy not by predicting charges but by memorizing the data.

regression_v2.py and **regression_v3.py**: the regression version 2 program plots the BMI vs the predicted values of the linear regression. The regression version 3 program plots the 'scatter plots' of each variable against the test data – the charges column.

random_forest.py: this program splits the data into test and training data, evaluates the model using K-Fold Cross Validation, and outputs the Random Forest model with its mean MAE and standard deviation MAE.

statistical_significance.py: this program collects various important statistics about the data and exports it to a text file named, "stats.txt".

individual_regressions.py: this program contains the various plots and models (Huber Regressor) for each variable against the Y variable (charges).

These files will be emailed on Friday to be supervised. I will try to include a "requirements.txt" file so that everyone could run that file and have all the required libraries. This will be a better option than to install every single library using "pip install <nameoflibrary>".

Also, I will postpone the Tableau analysis and instead will try to add a GUI interface or a website so that if someone doesn't have Tableau, they would still be able to see the results and play with the different plots.

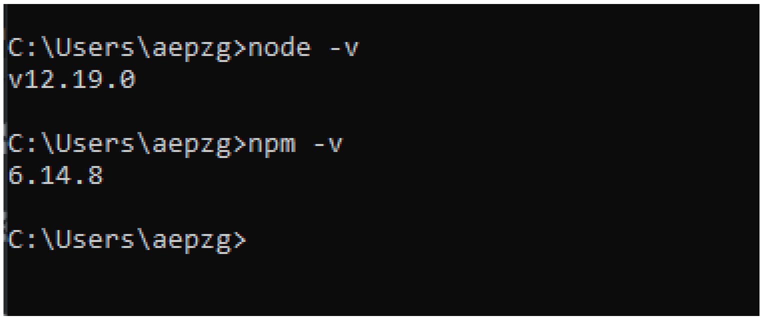
All the files will be "zipped" so that running them would be easier. Also attached are going to be the screenshots of all the pictures included in this report as well as the original dataset, 'insurance.csv'.

October 15th, 2020 – 9:00 AM EST – 3:00 PM EST

Today I decided to download the NodeJS and React frameworks for JavaScript in order to build the interactive dashboard – this will be a web app that everyone will be able to access.

I ran into some problems with the installation, however I ended up having all the dependencies and programs ready to go.

The problems I was getting had to do with Anaconda being installed as opposed to Python3. The solution was to start the installation of NodeJS from the beginning and installing "Chocolatey", which is a package manager for Windows – this is similar to Homebrew for MacOS. Then I proceeded to install the dependencies and check for the installation of the packages.



```
C:\Users\aepzg>node -v
v12.19.0

C:\Users\aepzg>npm -v
6.14.8

C:\Users\aepzg>
```

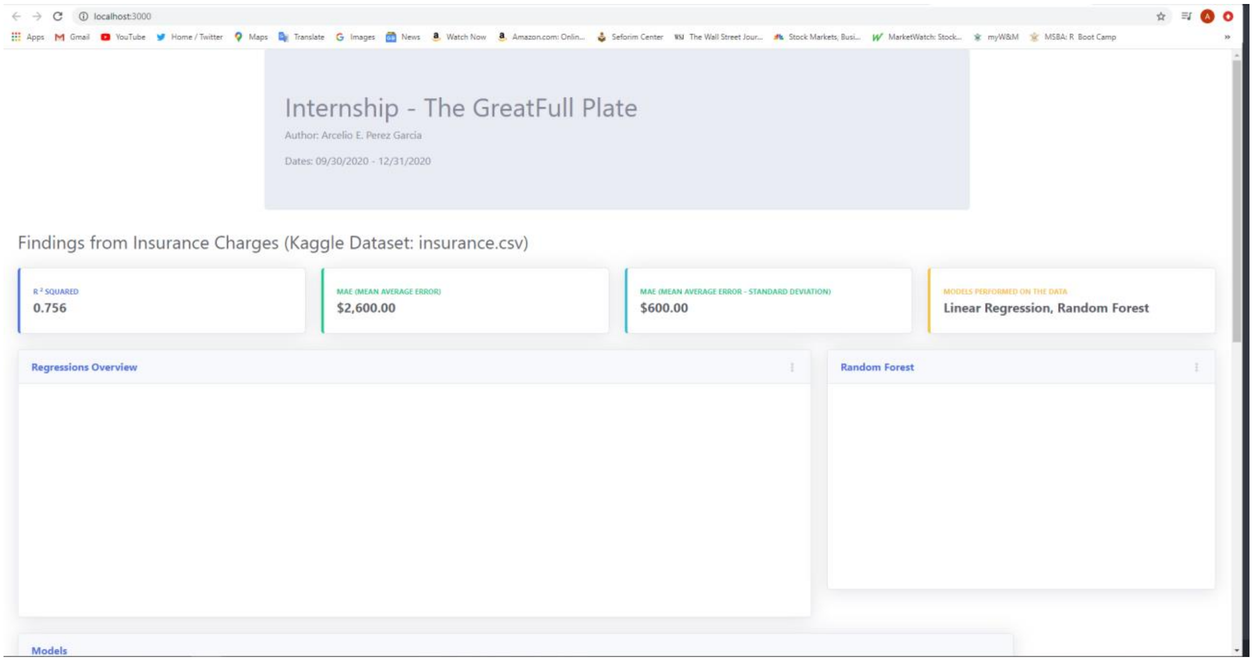
This command allows to check the version of the NodeJS framework

The plan for tomorrow is to continue working in the best way to build a dashboard and if needed more plots will be made. As previously mentioned, a comparison using R functions like glm (generalized linear regression), ggplt, and the randomForest package in R.

October 16th, 2020 – 9:00 AM EST– 3:00 PM EST

Today I started building the website using NodeJS and React. I took me a while because there were some problems with the dependencies and uploading the files to Github took some time.

Here are some pictures of the prototype of the website:



Internship - The GreatFull Plate

Author: Arcelio E. Perez Garcia

Dates: 09/30/2020 - 12/31/2020

Findings from Insurance Charges (Kaggle Dataset: insurance.csv)

R ² SQUARED
0.756

MAE (MEAN AVERAGE ERROR)
$2,600.00

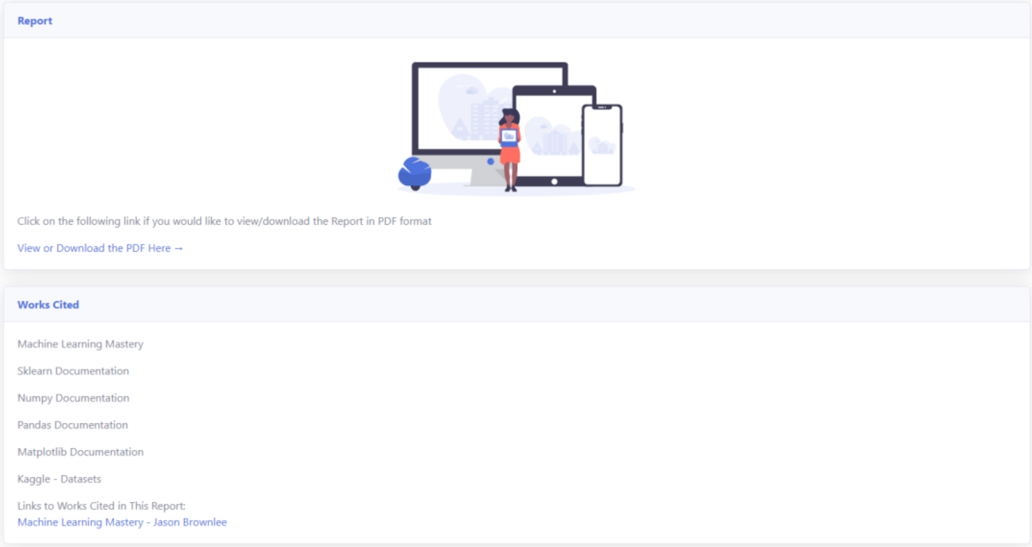
MAE (MEAN AVERAGE ERROR - STANDARD DEVIATION)
$600.00

MODELS PERFORMED ON THE DATA
Linear Regression, Random Forest

Regressions Overview

Random Forest

Models



Report

Click on the following link if you would like to view/download the Report in PDF format

View or Download the PDF Here →

Works Cited

Machine Learning Mastery

Sklearn Documentation

Numpy Documentation

Pandas Documentation

Matplotlib Documentation

Kaggle - Datasets

Links to Works Cited in This Report:
Machine Learning Mastery - Jason Brownlee

Internship The GreatFull Plate

The website will have the report in PDF so that anyone can view it or download it. It will also display the charts in an interactive way – I will plan to work next week in adding more functionality to the website and hopefully deploying it to the public.

As previously mentioned, the technology used is the React framework. Additionally, I found a Bootstrap template of a dashboard. This is the current template, but some modifications were made.

All the files will be available in my Github repository in the next couple of weeks.

October 19th, 2020 – 9:00 AM EST – 3:00 PM EST

API Key: AIzaSyAivQMxy5m0Ilgs4rpppk-lc8gJ8Ijr6Gs

Link to the Google Sheets data: https://docs.google.com/spreadsheets/d/1En2W9g0-oSvwQ9aTz9KWIcaUEFuRrgykgNEIiE2WTgI/edit?usp=sharing

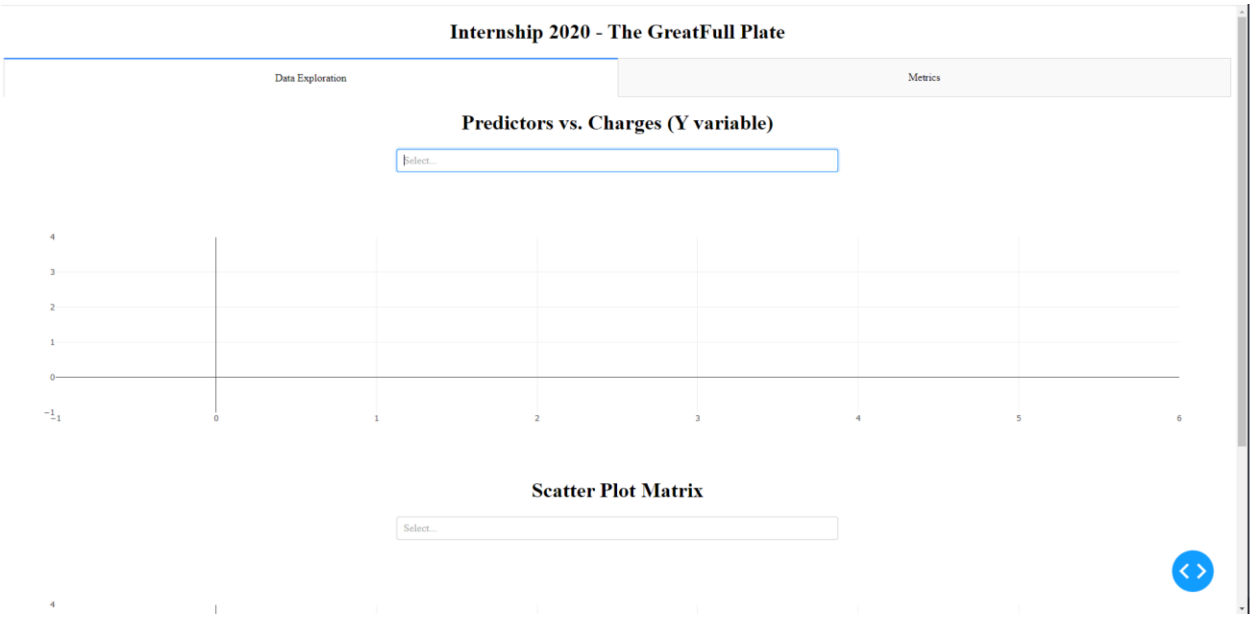
This connection will allow the future website to connect seamlessly with the data. I am still trying to deploy the website to Github pages; however, it is not displaying correctly. One of the reasons could be that using JavaScript dependencies are not that effective if the code is in Python.

A possible solution to this problem is to use the Flask framework for Python instead of React. Or even use R Pubs online. This week I will do my best to deploy a functional website that doesn't have the models yet, but that could serve as a data exploration website – with filters, maps, and charts.

October 20th, 2020 – 9:00 AM EST – 3:00 PM EST

Today I finished the design of the website – this time using Python instead of JavaScript. I found it to be easier to use, now the only aspects missing from the website are the interactive charts.

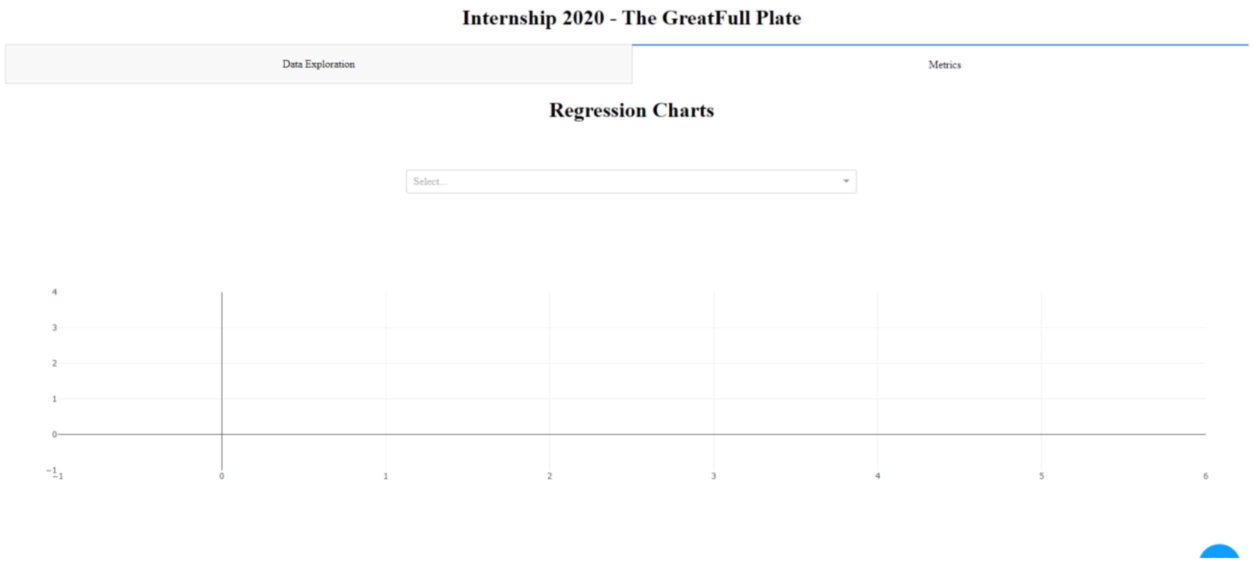
Here are some screenshots of the website in construction:


Internship 2020 - The GreatFull Plate

The difference between this website and the one using JavaScript is that the charts will be interactive. Dash offers several buttons that the user could use in order to zoom in, zoom out, and download the chart as a png file.

The data exploration tab includes each predictor against the Charges variable. The second graph is a scatter matrix of each variable against the Charges variable. Users will be able to change predictors and go the second tab which includes the machine learning models (linear regression and random forest).

Here is a screenshot of the second tab:


Internship 2020 - The GreatFull Plate

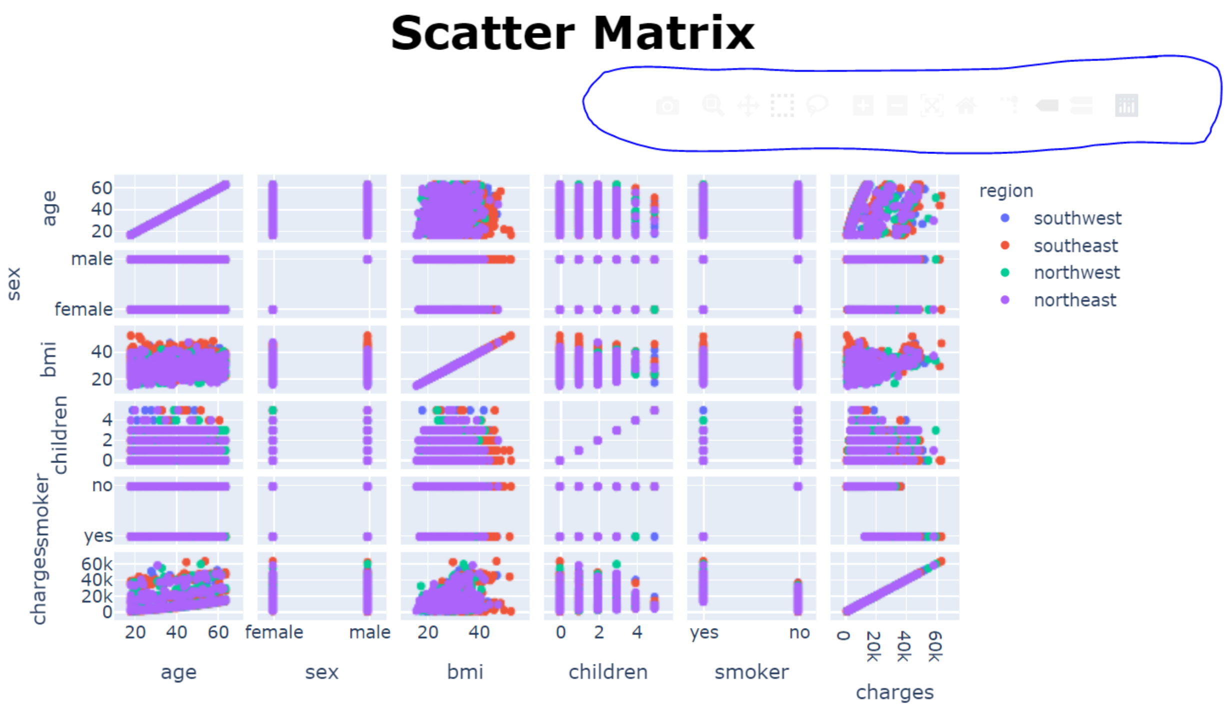
Users will be able to view their desired regression using the dropdown menu. Tomorrow I will need to add the random forest metrics and polish the design of the website.

October 21st, 2020 – 9:00 AM EST – 3:00 PM EST

Today I ran into some problems again with the dependencies – Python 3.9 is not working well with ‘pip’ so I had to use Anaconda to run the programs.

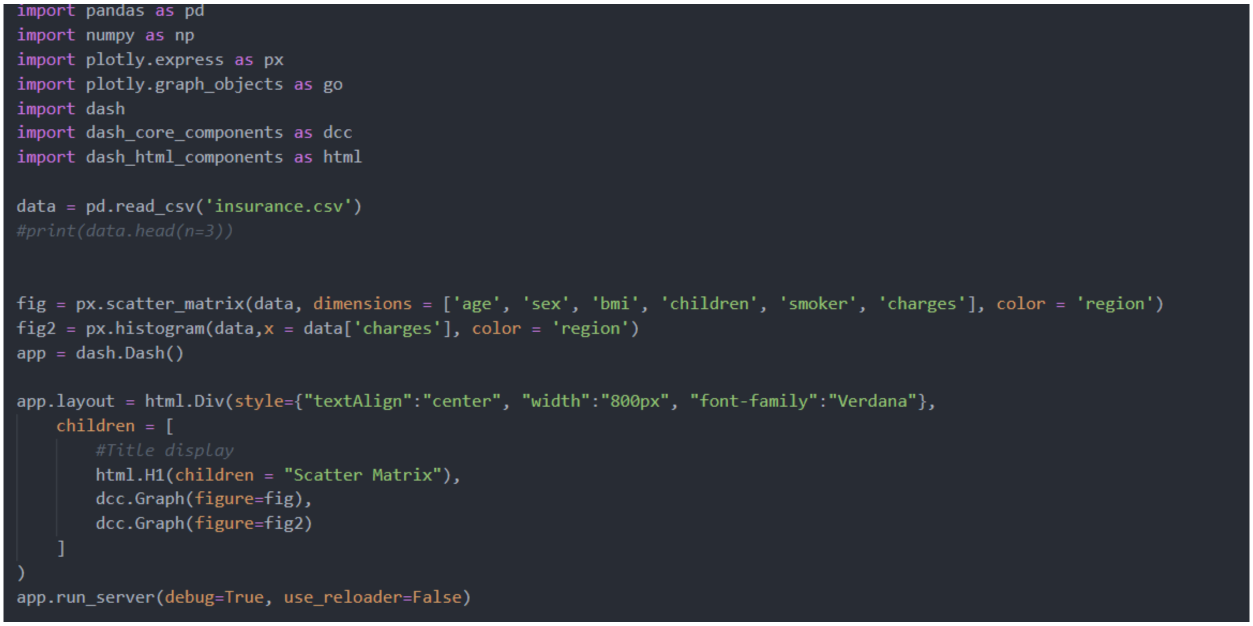
I finished the Data Exploration view of the website. Although I need to add those plots to the actual website with the data exploration tab.

Here is a screenshot of one of the plots:



The blue circle are all the plugins of the ‘plotly’ package. They include zoom (in and out), download, as well as expanding the image.

The code to produce this image is:



```
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
import dash
import dash_core_components as dcc
import dash_html_components as html

data = pd.read_csv('insurance.csv')
#print(data.head(n=3))


fig = px.scatter_matrix(data, dimensions = ['age', 'sex', 'bmi', 'children', 'smoker', 'charges'], color = 'region')
fig2 = px.histogram(data,x = data['charges'], color = 'region')
app = dash.Dash()

app.layout = html.Div(style={"textAlign":"center", "width":"800px", "font-family":"Verdana"},
    children = [
        #Title display
        html.H1(children = "Scatter Matrix"),
        dcc.Graph(figure=fig),
        dcc.Graph(figure=fig2)
    ]
)
app.run_server(debug=True, use_reloader=False)
```

Where dash is the dashboard package and plotly is the especial plotting library. The two plots included so far in the website are a scatter matrix of all the variables (colored by the region), and a histogram of the insurance charges (colored by the region).

Depending on the feedback I could add more plots or leave these two plots as the basis of the website.

The only pending part so far is the models tab with the visualizations that go with it.

October 22nd, 2020 – 12:00 PM EST – 6:00 PM EST

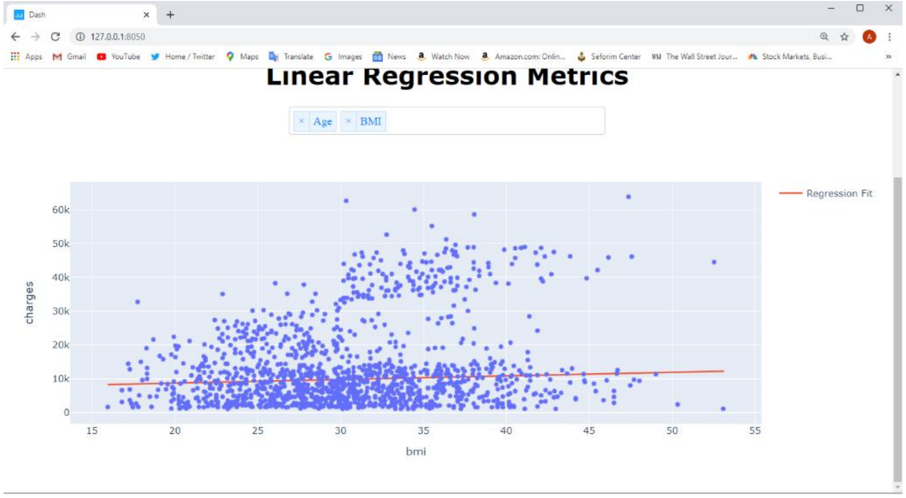
***CORRECTION TO YESTERDAY'S UPDATE THAT SAID '12:00 AM'**

I started working today at 12 because I was not feeling well. However, I finished adding the machine learning models into the website.

Website url: https://my-internship-app.herokuapp.com/

The additional plot in the 'Performance Metrics' tab include every regression (every predictor against the charges variable).

Screenshot:



The next addition is the Random Forest model – box plot and metrics. And adding the metrics for each of the models performed.

Another addition is the to include at least 20 rows from the dataset.

This deployment used Heroku, Git, and Python. I wanted to use Github, but it does not support the deployment of Dash apps yet.

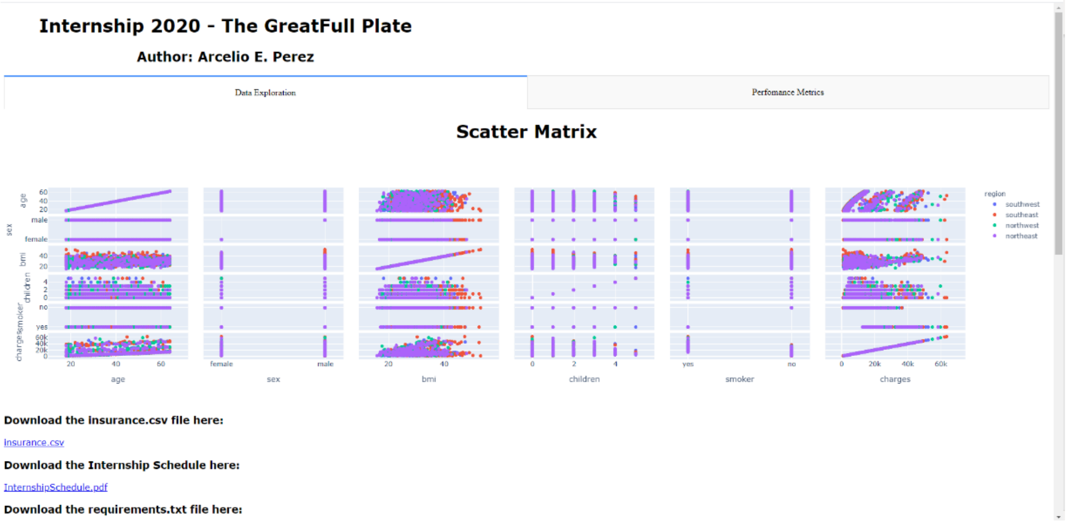
October 23rd, 2020 – 9:00 AM EST – 3:00 PM EST

Today I added a markdown to the website and the files to download. However, if I deploy the website to Heroku the files could be downloaded but they don't work. If I instead used the app offline, I can download the files with no issues.

I will try to use Github to add these files to their own URL so that I can link them inside the Markdown instead of adding more folders for deployment.

Next week I will keep working on the website and add more functionality and explanations of the models.

Screenshots of the website so far:



Internship 2020 - The GreatFull Plate

Author: Arcelio E. Perez

Scatter Matrix

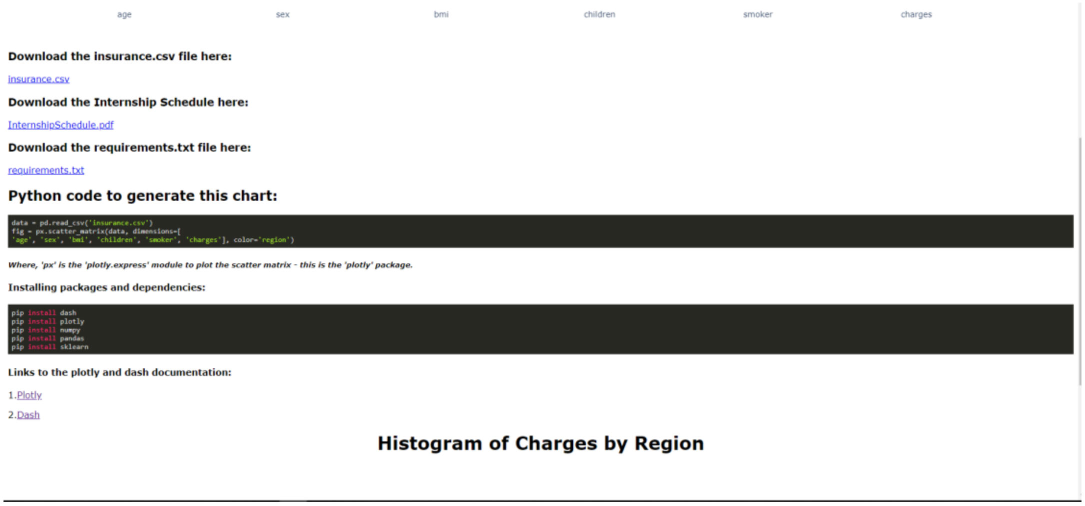
Download the insurance.csv file here:

insurance.csv

Download the Internship Schedule here:

InternshipSchedule.pdf

Download the requirements.txt file here:



Download the insurance.csv file here:

insurance.csv

Download the Internship Schedule here:

InternshipSchedule.pdf

Download the requirements.txt file here:

requirements.txt

Python code to generate this chart:

```
data = pd.read_csv('insurance.csv')
fig = px.scatter_matrix(data, dimensions=[
'age', 'sex', 'bmi', 'children', 'smoker', 'charges'], color='region')
```

Where, 'px' is the 'plotly.express' module to plot the scatter matrix - this is the 'plotly' package.

Installing packages and dependencies:

```
pip install dash
pip install plotly
pip install numpy
pip install pandas
pip install sklearn
```

Links to the plotly and dash documentation:

1.Plotly

2.Dash

Histogram of Charges by Region

URL: https://my-internship-app.herokuapp.com/

October 26th, 2020 – 9:00 AM EST – 3:00 PM EST

Made some changes to the website – layout and fixed the scatter plot matrix plot. Also added some content to the website.

Things to work on this week:

Add more functionality to the second tab

Add the findings from the random forest model

Add a tab that explains how to use dash and plotly

Explain how to use Git, Github, and Heroku

Website URL: https://my-internship-app.herokuapp.com/

October 27th, 2020 – 9:00 AM EST – 3:00 PM EST

Today I decided to take a demo (GIF) using the Heroku website. The reason is that I realized that Heroku only gives each app a free ***dyno*** with only **550 hours per month**. I do not want this website to stop working so I will take a short video tour of the app and show the basic app functionalities.

In order to prevent a future crash of the website, I will use Github to host a ***clone*** website – basically another website that has the basic functionality of the website that is now in use.

I will also document the code so that it could be replicated – and changed if needed.

Links to Github – website with the Heroku demo: https://arcelioeperez.github.io/dash-app/

Links to the **gh-pages** branch in the **dash-app** repository: https://github.com/arcelioeperez/dash-app/tree/gh-pages

I need to add all the files, however, since there are a lot of files, I will add them once I complete the internship. All the files could be downloaded using the **.zip** and/or **tar.gz** buttons on the Github website.