LaShanni Butler
Flatiron School – Data Science
Module 3 blog
5/28/19

Car Insurance Predictive Modeling

Atlas I'm wrapping up another project for my data science course at Flatiron. This time we're to undertake predictive modeling and machine learning as we conclude Module 3 in the program. Now I find Machine Learning very interesting, since it appeals to my inquisitive nature. I really like that I can apply this technique to any future job(s) I'll have (especially since Machine Learning is widely used in health care and medical data). So for this blog, I'm going to describe my workflow for this project (and hopefully it'll make sense)!

This project was slightly different than previous ones. Instead of being given a data set, we had to seek out and find one on the interwebs! So right off the bat, I was flustered because there were so many data sets to choose from online. I found many interesting data sets, but I ultimately chose one from Kaggle, which was about Car Insurance Cold Calling. This data set fit the requirements such as a minimum of 1000 rows, at least 10 predictor columns, complexity, etc. However, what I really liked about this data set were the simplicity and it was somewhat clean. So, this really saved me a lot of time. Now I know in the real world, we don't get this option of working with squeaky clean data. But for the purposes of this project (and my limited time constraints) this worked perfectly for my needs. Additionally, since my background is scientific in nature, it was very nice working with a non-scientific data set.

Once my data set was approved, the next requirement was my problem statement. I could either ask the initial problem or do more of an ad-hoc approach, and look at the data first, then ask the question. I decided to go with the ad-hoc approach. I thought the latter approach would make more sense, since I was working with data in an unfamiliar field. Feel free to continue reading for my problem statement.

I opened my handy dandy Jupyter Notebook, imported the necessary libraries and data set and started my Machine Learning undertaking. Also, this project as with my previous projects did employ the OSEMN framework. A quick reminder that OSEMN refers to:

- O — Obtaining our data
- S — Scrubbing / Cleaning our data
- E — Exploring / Visualizing our data will allow us to find patterns and trends
- M — Modeling our data will give us our predictive power as a wizard
- N — Interpreting our data

You can follow along knowing this was the methodology I used.

As I mentioned before, I was attracted to this data set because it was already somewhat "clean." This however did not mean there was no scrubbing involved. After running df.head(), df.tail(), etc. I ran df.info() to get an idea of my columns. I had 10 integer columns and 8 object columns, as well as some NaN values. Before I tinkered with the NaNs, I wanted to see if I had any outliers, which would \ skew my analyses. One column, Balance, did have an extreme outlier, so I did eliminate that using the df.drop() method. Then I circled back to my null values, which were the Job, Education, Communication and Outcome columns. Now Job and Education had much fewer missing values than Communication and Outcome. I found a cool feature from StackOverflow.com on the front fill or forward fill method for missing values. This method propagates the last valid observation forward. The Communication and

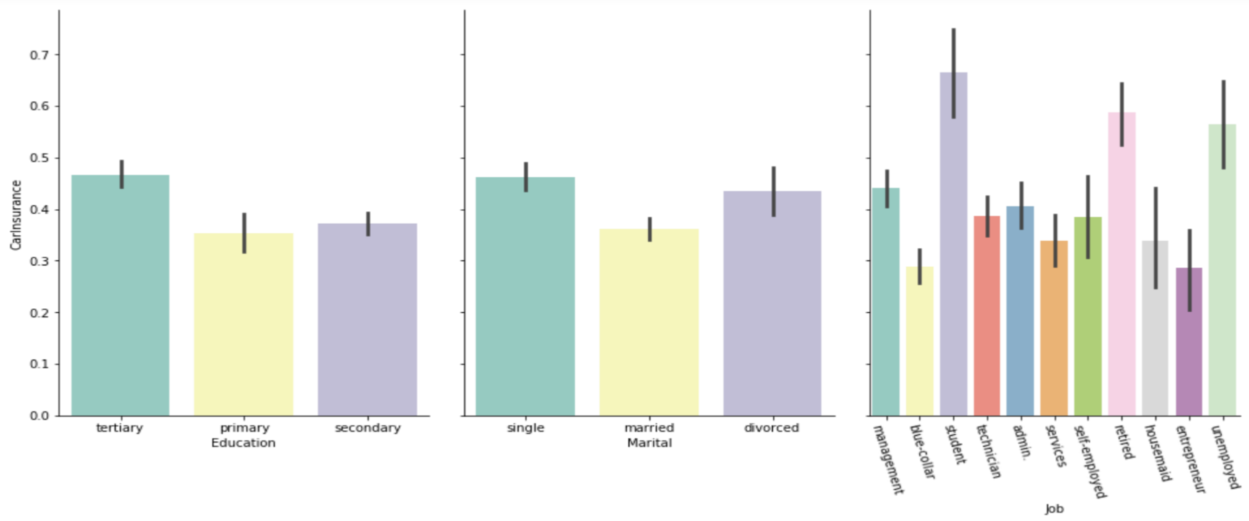
LaShanni Butler
Flatiron School – Data Science
Module 3 blog
5/28/19

Car Insurance Predictive Modeling

Outcome columns had the df.fillna() method applied to them for their missing values. Once I used these techniques another check of missing values using df.isnull().sum() was done and…voila all the missing values were filled.

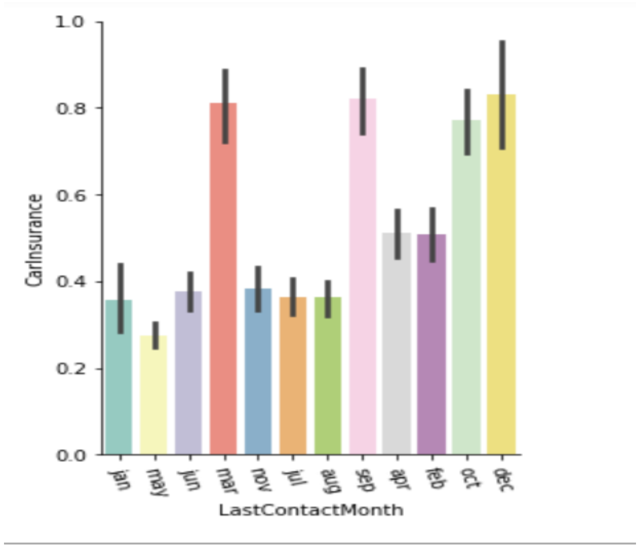
Next step was the exploratory data analysis (EDA). Now I really enjoy this part, because it allows my visual curiosities to be satisfied with cool graphs and images. I generated some histograms but nothing really stood out at me. I did notice many columns (or features) were binary. This would be important and helpful for my predictive modeling later on. A visualization that would be more helpful is a heatmap, so I did this next. Heatmaps are great at showing correlations, and I needed to understand the relationships for my ad-hoc problem statement. Upon running the heatmap/correlation, I saw that the Car Insurance feature had many positive correlations with the other features. So that allowed me to figure out my problem statement. I wanted to know ***which features in my data set would determine whether a potential customer would buy car insurance***?

Since I now was interested in the Car Insurance feature, it was time to do more EDA. I created some bar plots with error bars with certain features on the x-axis and Car Insurance along with y-axis. The graphs showed me some interesting trends like potential customers with advanced degrees are likely to purchase car insurance, and single people are also likely to buy insurance. Also, those who are students, retired and unemployed purchased the most insurance. Lastly, car insurance sells appear to peak in March, Sept, Oct, and Dec.



LaShanni Butler
Flatiron School – Data Science
Module 3 blog
5/28/19

Car Insurance Predictive Modeling



Ok so that's cool, but really doesn't mean too much without predictive modeling. That brought me to the next step (and the meat of this project - Modeling).

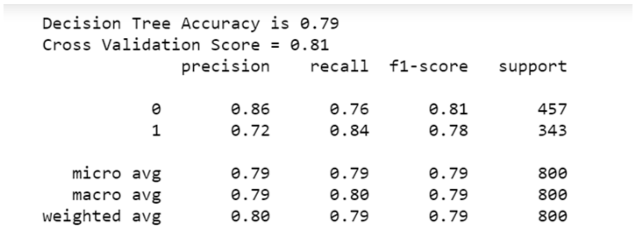
Now this is where I started to pull my hair out, because cleaning the data for predictive modeling was challenging. With this data set, the Age and Balance columns were continuous. I discovered that they needed to be binned. I found some examples in other python codes where others binned columns into 5 segments using pd.qcut(). I also had some object variable columns, Call Start and Call End, which would be problematic in my predictive modeling. Both these columns would have to be converted into the datetime function, subtracted and then binned. Not fun but had to be done. I also had to bin more columns and convert the categorical columns to dummy values. My rudimentary understanding of this is switching to dummy values is required so your model will be able to understand these values. Using the pd.get_dummies() function, also known as one-hot encoding creates a new column for each category. For example, if the column has 10 unique values, this will create 10 variables, and encode the value as a boolean (1 or 0). Now all of this was quite foreign to me, but luckily there were loads of examples online. Once I did this (which took me a while to figure out) I ran df.columns and saw all the new columns changes were in effect.

Finally, I was ready to do some predictive modeling with my data! I setup my train/test split data and went to work. Since I was focusing on classification for my predictive modeling, I choose to run the following: K-Nearest Neighbors (kNN), Logistic Regression, Support Vector Machine (SVM), Decision Tree and Random Forest. These models are examples of ***Supervised*** machine learning. Supervised machine learning is a method that enables the machine to classify and predict objects, problems or situations based on labeled data. With this type of model, your data is labeled, direct feedback is given, and the machine predicts the output.

LaShanni Butler
Flatiron School – Data Science
Module 3 blog
5/28/19

Car Insurance Predictive Modeling

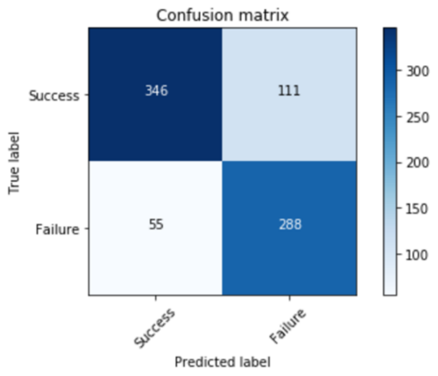
With each predictive model, confusion matrices would be needed. Confusion matrices are helpful because they assess the accuracy of each model used. Basically, they're a summary of prediction results on your classification problem. The output generated from your code will be a cool looking square:



```
Decision Tree Accuracy is 0.79
Cross Validation Score = 0.81
              precision    recall  f1-score   support

           0       0.86      0.76      0.81       457
           1       0.72      0.84      0.78       343

   micro avg       0.79      0.79      0.79       800
   macro avg       0.79      0.80      0.79       800
weighted avg       0.80      0.79      0.79       800
```



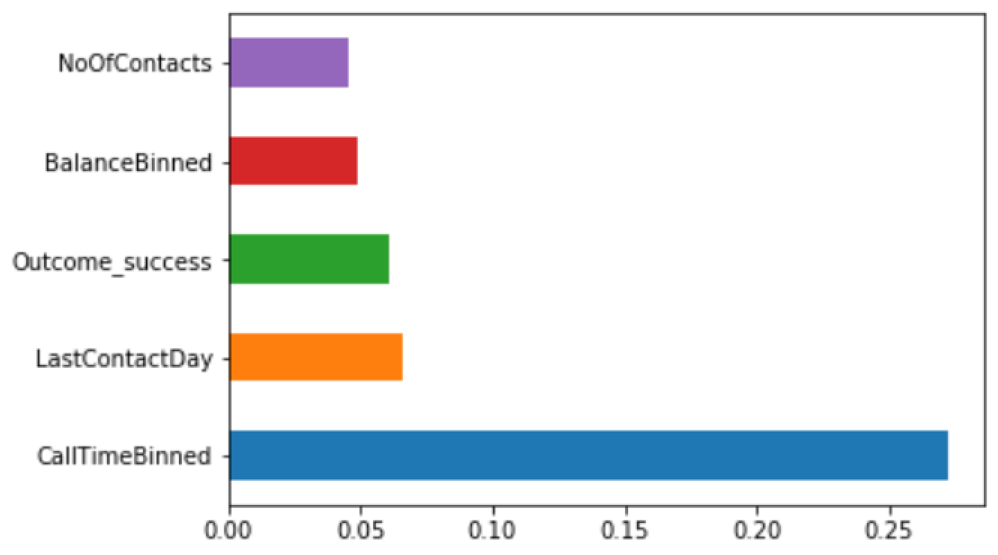
For instance, this above confusion matrix was generated for my Decision Tree model. It looks very similar to a heatmap. The gist of this bad boy is you count all the success and failure values in each square (mine totaled 800). I'm interested in the percentage of the true positive and true negative values / the total count. So that would be the values in the darkest blue squares (346+288 / 800 = 0.79). If you multiply 0.79 x 100, this would generate 79%. So, my confusion matrix for this model shows an accuracy of 79%. For the models, my confusion matrices accuracy were:

- K-Nearest Neighbors (kNN): 74%
- Logistic Regression: 81%
- SVM: 79%
- Decision Tree: 79%
- Random Forest: 83%

LaShanni Butler
Flatiron School – Data Science
Module 3 blog
5/28/19

Car Insurance Predictive Modeling

So based on my predictive models, Random Forest seemed to be the best predictive model as it had the highest accuracy. Additionally, I wanted to know which features in this data set were the most important. This would help with the cold calling efforts as it would most likely drive the most sales. I used the ExtraTreesClassifer in Sklearn and generated the following graph of the top 5 features:



The graph showed that the Call Time (i.e. – time on the phone with potential customers) was the top feature in this data set. Last Contact Day also showed that potential customers who were contacted regularly seemed more likely to purchase car insurance.

Overall, my interpretation of the data showed that Call Time influenced the likelihood of a potential customer buying car insurance. In my recommendations to this fictional company, I would strongly urge them to focus on this. Additionally, as I stated previously, customers with advanced degrees are likely to purchase car insurance, and single people are also likely to buy insurance. Also, those who are students, retired and unemployed purchased the most insurance. Moreover, car insurance sells appear to peak in March, Sept, Oct, and Dec. This almost seemed to show a quarterly pattern, so maybe there is something to buying insurance with the change of seasons. I would recommend this fictional company conducting car insurance sells to strongly focus on these demographics and trends. Lastly, Random Forest modeling showed to have the highest level of accuracy. And with all that being said, this concluded my module 3 project! It was a beast, but very enjoyable once I got the hang of the nuances involved with predictive modeling and machine learning.