W207 Final Project: Santander Product Recommendations

Ankith Gunapal, Prajakta Pandharkar, Lisa Barceló

Introduction

- We are provided with 1.5 years of customers behavior data from Santander bank to predict what new products customers will purchase.
- The data has monthly records of products a customer has, such as "credit card", "savings account", etc.
- You will predict what **additional** products a customer will get in the last month, 2016-06-28

Data Cleaning

- Very large CSV files, we had to import only some of the rows at the beginning
- Because we were supposed to predict June purchases, we decided to limit our training data to the month of June, for the 23 products listed
- We had to clean up some NaN values as well

canal_entrada, cod_prov, nomprov, and segmento contain NaN values.

filter all the rows containing NaN.

Including canal_entrada is causing issues. The Model throws an error. Not sure what the outlier is

In [16]:
```
g_no_nan = g[~g['segmento'].isnull().values & ~g['renta'].isnull().values  &
             ~g['cod_prov'].isnull().values & ~g['nomprov'].isnull().values]
labels_no_nan = labels[~g['segmento'].isnull().values & ~g['renta'].isnull().values &
                       ~g['cod_prov'].isnull().values & ~g['nomprov'].isnull().values]
```

In [17]:
```
data = g_no_nan[['ind_empleado','ncodpers','age','sexo','ind_nuevo','antiguedad','indrel','indrel_1mes',
         'tiprel_1mes','indresi','indext','indfall','ind_actividad_cliente','cod_prov','nomprov','renta','segmento']]
type(data)
```

Feature Engineering

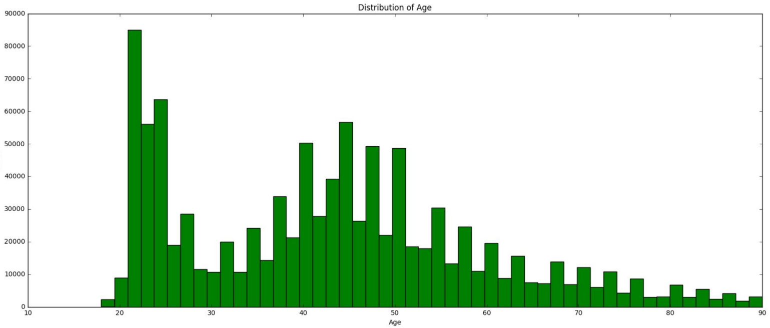
- Since features were categorical values, we used a Vectorizer to convert features into a matrix
- We only used some features, avoiding those with a lot of NaN values, and selecting those we felt would be more relevant to our models, such as age, seniority, and gender

We see that there are very few customers who are below the age of 20 and above age 90. Hence, we can transform the ages of these customers to be in the valid region for the model to perform better

In [95]:
```
f.loc[f.age < 18,"age"]  = f.loc[(f.age >= 18) & (f.age <= 30),"age"].mean(skipna=True)
f.loc[f.age > 90,"age"] = f.loc[(f.age >= 30) & (f.age <= 90),"age"].mean(skipna=True)
f["age"].fillna(f["age"].mean(),inplace=True)
f["age"]                    = f["age"].astype(int)
```

Transformed Age

In [96]:
```
f["age"]   = pd.to_numeric(f["age"], errors="coerce")
ax = plt.figure(figsize=(20,8))
n,bins,patches = plt.hist(f["age"].dropna(), 50, facecolor='green')
plt.xlabel("Age")
plt.ylabel("Count")
plt.title("Distribution of Age")
```



renta (income) should be an important feature. Hence, we need to rectify the data if renta has many missing values.

In [97]:
```
f.renta.isnull().sum()
```
Out[97]: 18284

A lot of renta entries are missing. We could try replacing NA with the median of renta

In [98]:
```
print f.renta.median()
```
89610.21

In [100]:
```
f.loc[f.renta.isnull(), "renta"] = f.renta.median()
```

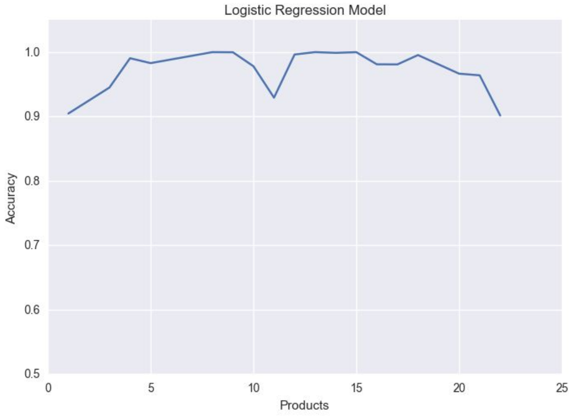
In [101]:
```
f.renta.isnull().sum()
```
Out[101]: 0

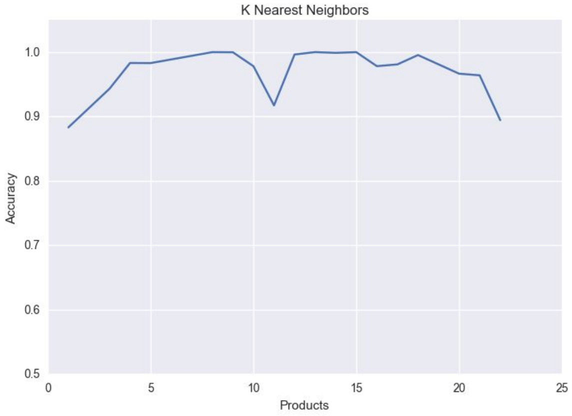
Experimenting with Different Models

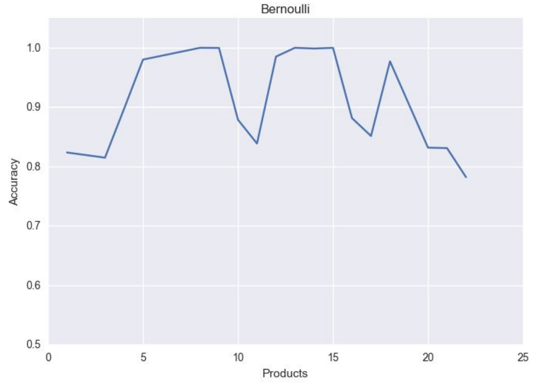
- We had success with Logistic Regression, K Nearest Neighbors, and Bernoulli
- Multinomial Bayes was not as successful, nor was Random Forests

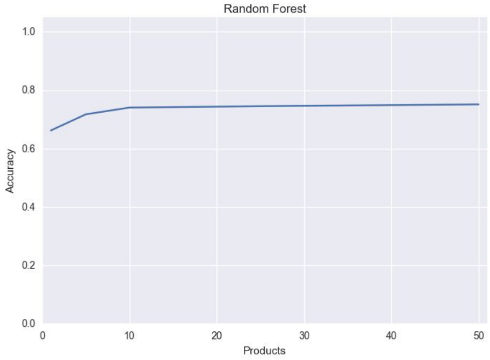
Model Tables			
Single Class	Logistic Regression 0.97	Bernoulli 0.91	Multinomial Bayes 0.70
Multi Class	K Nearest Neighbors 0.97	Random Forests 0.75	

Experimenting with Different Models



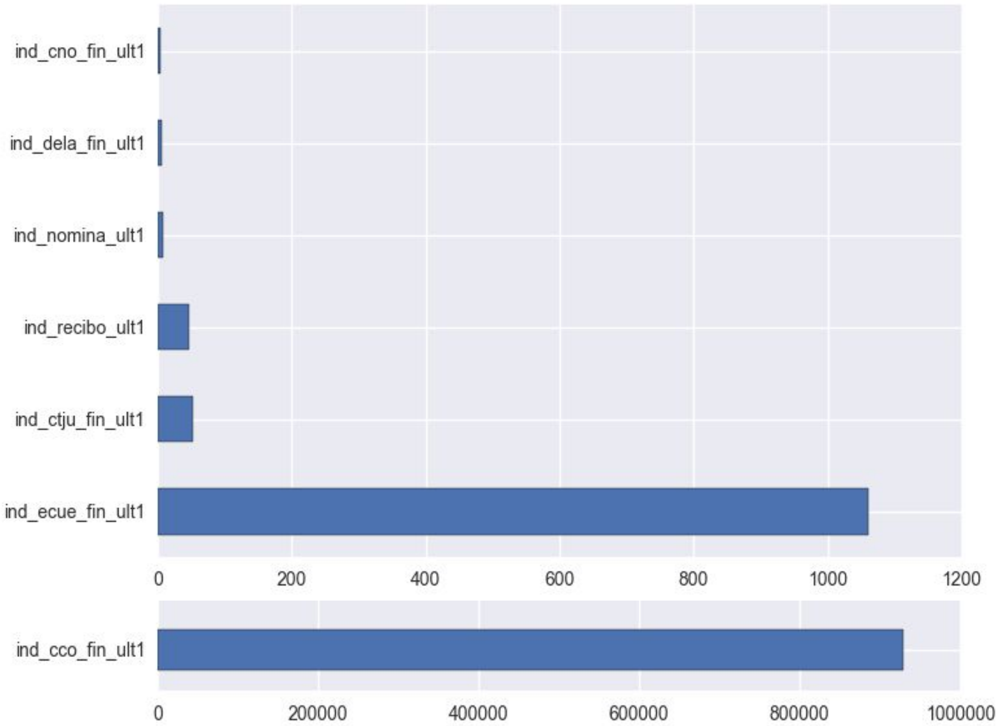






Model Recommendation

- There were about 900 values for which there were blank values, so we filled them in with the most common product
- We ended up using K Nearest Neighbors to predict the feature labels for our test data



Conclusions

- We were able to get a list of the products that users would be most likely to purchase in June
- Compared to when we used Logistic Regression, our score went up significantly by using K Nearest Neighbors--and we went up 94 places in the Kaggle Leaderboard