Analysis of Marketing from the Banco de Portugal

The data used in this project comes from a marketing campaign by the Banco de Portugal to encourage customers to subscribe to a term deposit. It includes attributes about the customer, how the marketing was conducted, relevant economic indicators, and whether the offer was accepted or declined.

In [1]:
```
import pandas as pd
import statsmodels.formula.api as smf
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
%matplotlib inline
```

In [2]:
```
df= pd.read_excel([redacted])
```

In [3]:
```
df.head()
```

Out[3]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	em
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	

5 rows × 21 columns

In [4]:
```
df = df[df != 'unknown']
df.dropna(inplace=True)
#Deleting cells with unknown information from data
```

```
C:\Users\Woyte\Anaconda3\lib\site-packages\pandas\core\ops.py:1649: FutureWarning: elementwise comparison
failed; returning scalar instead, but in the future will perform elementwise comparison
  result = method(y)
```

In [5]:
```
dummy=pd.get_dummies(df['default'])
dummy2=pd.get_dummies(df['housing'])
dummy3=pd.get_dummies(df['loan'])
dummy4=pd.get_dummies(df['y'])
df= pd.concat([df, dummy], axis=1)
df= df.drop('no',axis=1)
df['Ydefault']=df['yes']
df=df.drop('yes', axis=1)
df= pd.concat([df, dummy2], axis=1)
df= df.drop('no',axis=1)
df['Yhousing']=df['yes']
df=df.drop('yes', axis=1)
df= pd.concat([df, dummy3], axis=1)
df= df.drop('no',axis=1)
df['Yloan']=df['yes']
df=df.drop('yes', axis=1)
df= pd.concat([df, dummy4], axis=1)
df= df.drop('no',axis=1)
df['accepted']=df['yes']
df=df.drop(['default', 'housing', 'loan', 'y', 'yes'], axis=1)
df.head()
#Creating a lot of dummy variables from categorical data
```

Out[5]:

	age	job	marital	education	contact	month	day_of_week	duration	campaign	pdays	...	poutcome	emp.var.rate	cons.pri
0	56	housemaid	married	basic.4y	telephone	may	mon	261	1	999	...	nonexistent	1.1	
2	37	services	married	high.school	telephone	may	mon	226	1	999	...	nonexistent	1.1	
3	40	admin.	married	basic.6y	telephone	may	mon	151	1	999	...	nonexistent	1.1	
4	56	services	married	high.school	telephone	may	mon	307	1	999	...	nonexistent	1.1	
6	59	admin.	married	professional.course	telephone	may	mon	139	1	999	...	nonexistent	1.1	

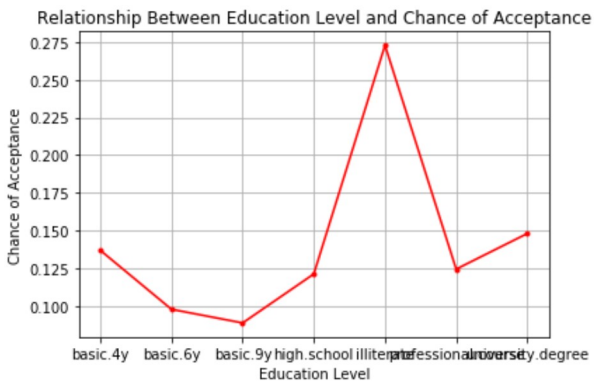
5 rows × 21 columns

In [6]:
```
#Running a simple analysis to make sure everything is correct
group=df['accepted'].groupby([df['education']]).mean()
group
```

Out[6]:
```
education
basic.4y                0.136975
basic.6y                0.097912
basic.9y                0.088868
high.school             0.121314
illiterate              0.272727
professional.course     0.124508
university.degree       0.148098
Name: accepted, dtype: float64
```

In [7]:
```
plt.grid(True)
plt.xlabel('Education Level')
plt.ylabel('Chance of Acceptance')
plt.title('Relationship Between Education Level and Mean Chance of Acceptance')
plt.plot(group,c= 'red', marker='.')
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0x200da7bb390>]
```



Results

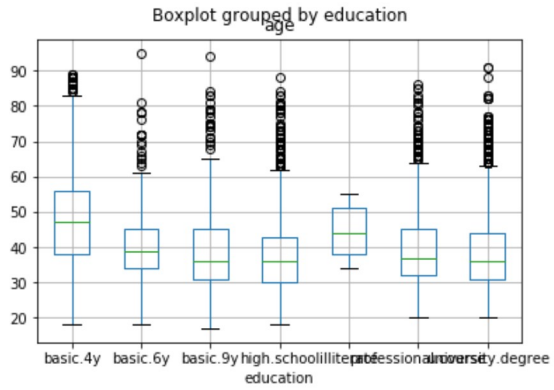
As the data shows, there is an interesting relationship between education and the mean chance of acceptance of the bank's offer. Generally, from illiteracy to some high school education, the more education they have, the less likely the person is of accepting the bank's offer. This relationship reverses when someone graduates from high school though, slowly gaining in likelihood to accept the bank's offer the more education they receive.

Overall, the illiterate were the most likely to accept a loan, with a mean chance of 27%, and those with nine years of education are the least likely, only a 9% chance.

ANOVA Comparing Ages to Customer Education

In [8]:
```
df.boxplot('age', by= 'education')
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x200db004048>
```



In [9]:
```
mod = smf.ols('age ~ education', data=df).fit()
anova_table = sm.stats.anova_lm(mod, typ=2)
anova_table
```

Out[9]:

	sum_sq	df	F	PR(>F)
education	2.123928e+05	6.0	354.573796	0.0
Residual	3.043065e+06	30481.0	NaN	NaN

In [10]:
```
pair_t = mod.t_test_pairwise('education')
pair_t.result_frame
```

Out[10]:

	coef	std err	t	P>|t|	Conf. Int. Low	Conf. Int. Upp.	pvalue-hs	reject-hs
basic.6y-basic.4y	-7.710521	0.337376	-22.854371	1.218471e-114	-8.371793	-7.049250	0.000000e+00	True
basic.9y-basic.4y	-9.453235	0.255529	-36.994754	4.280600e-293	-9.954083	-8.952388	0.000000e+00	True
high.school-basic.4y	-10.405884	0.234339	-44.405310	0.000000e+00	-10.865198	-9.946570	0.000000e+00	True
illiterate-basic.4y	-3.473071	3.019576	-1.150185	2.500767e-01	-9.391566	2.445424	4.376150e-01	False
professional.course-basic.4y	-8.412774	0.255053	-32.984432	1.816088e-234	-8.912688	-7.912859	0.000000e+00	True
university.degree-basic.4y	-9.420885	0.227015	-41.498943	0.000000e+00	-9.865844	-8.975926	0.000000e+00	True
basic.9y-basic.6y	-1.742714	0.308582	-5.647486	1.642591e-08	-2.347548	-1.137880	1.971109e-07	True
high.school-basic.6y	-2.695363	0.291277	-9.253593	2.308216e-20	-3.266279	-2.124447	0.000000e+00	True
illiterate-basic.6y	4.237450	3.024528	1.401029	1.612156e-01	-1.690750	10.165651	4.098656e-01	False
professional.course-basic.6y	-0.702252	0.308188	-2.278649	2.269482e-02	-1.306314	-0.098191	1.594399e-01	False
university.degree-basic.6y	-1.710364	0.285419	-5.992475	2.089880e-09	-2.269796	-1.150931	2.716844e-08	True
high.school-basic.9y	-0.952649	0.190565	-4.999080	5.792344e-07	-1.326164	-0.579134	5.792329e-06	True
illiterate-basic.9y	5.980164	3.016495	1.982488	4.743361e-02	0.067709	11.892619	2.529124e-01	False
professional.course-basic.9y	1.040462	0.215528	4.827503	1.389244e-06	0.618018	1.462905	1.250313e-05	True
university.degree-basic.9y	0.032350	0.181483	0.178256	8.585233e-01	-0.323364	0.388065	8.585233e-01	False
illiterate-high.school	6.932813	3.014773	2.299613	2.147686e-02	1.023731	12.841895	1.594399e-01	False
professional.course-high.school	1.993110	0.189926	10.494154	1.016663e-25	1.620848	2.365373	0.000000e+00	True
university.degree-high.school	0.984999	0.150185	6.558552	5.520139e-11	0.690629	1.279369	7.728196e-10	True
professional.course-illiterate	-4.939703	3.016454	-1.637586	1.015185e-01	-10.852079	0.972674	3.483166e-01	False
university.degree-illiterate	-5.947814	3.014213	-1.973256	4.847543e-02	-11.855797	-0.039830	2.529124e-01	False
university.degree-professional.course	-1.008111	0.180812	-5.575465	2.489582e-08	-1.362510	-0.653712	2.738540e-07	True

Results

As the above tables show, there is a very distinct statistical difference between the data from the varying groups. The first ANOVA test, which tested the model as a whole, came back with a p-value of less than .01, meaning that there was a statistical difference even at a 99% confidence level. Running the test on each individual grouping for Education came back showing that thirteen out of the twenty one relationships, or 62% of them, were statistically different from the others.

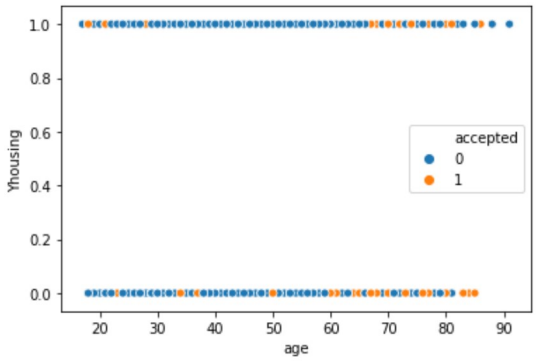
K-Nearest Neighbors

Let's a new candidate came in the ways of a 45-year-old man withno housing loan. Would they be expected to accept the bank's offer?

In [11]:
```
X = pd.DataFrame(data = df ,columns = ['age', 'Yhousing'])
X = X[['age', 'Yhousing']]
y= df['accepted']
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
```

In [12]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean')
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
sns.scatterplot(
    x='age',
    y='Yhousing',
    hue='accepted',
    data=X_test.join(y_test, how='outer'))
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x200da088898>
```



In [13]:
```
confusion_matrix(y_test, y_pred)
```

Out[13]:
```
array([[6552,   99],
       [ 925,   46]], dtype=int64)
```

In [14]:
```
x2 = [[45, 0]]
pred = knn.predict(x2)
print("X=%s, Predicted=%s" % (x2[0], pred[0]))
```

```
X=[45, 0], Predicted=0
```

Results

With a true positive and a true negative of 6,552 and 46 instances respectively, that means the models accurate results were 6,598. Adding on the false positives and false negatives, and our model had an accuracy of 6,598/7,622 or 86.5%.

As for the new instance itself. If there is a 45-year-old person without a housing loan, they are not expected to accept the bank's offer.

Building a Classification Tree Model

Another new person is added to the marketing data. This time it is a 50 year-old woman who has never defaulted, has a housing loan, but does not have a regular loan.

In [15]:
```
X = pd.DataFrame(data = df, columns = ['age', 'Yhousing', 'Ydefault', 'Yloan'])
y= df['accepted']
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
#Splitting the data set for prediction and setting up variables.
```

In [16]:
```
clf = DecisionTreeClassifier(max_depth = 1, random_state = 0)
clf.fit(X_train, y_train)
```

Out[16]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=1,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=0, splitter='best')
```

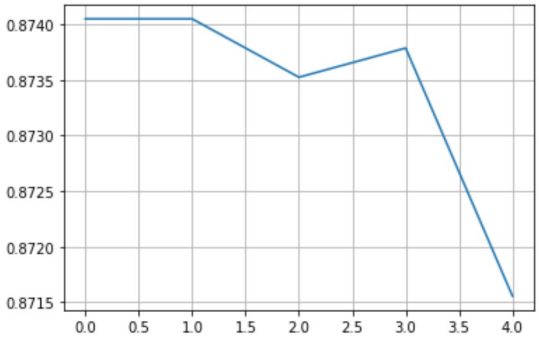
In [17]:
```
#Testing accuracy of model.
score = clf.score(X_test, y_test)
score
```

Out[17]:
```
0.874048806087641
```

In [18]:
```
#Creating function to test out other possible maximum depths of the model and their accuracy
max_depth_range = list(range(1, 6))
accuracy = []
for depth in max_depth_range:

    clf = DecisionTreeClassifier(max_depth = depth,random_state = 0)
    clf.fit(X_train, y_train)
    score = clf.score(X_test, y_test)
    accuracy.append(score)
plt.grid(True)
plt.plot(accuracy)
```

Out[18]:
```
[<matplotlib.lines.Line2D at 0x200dc82a128>]
```



In [19]:
```
accuracy
```

Out[19]:
```
[0.874048806087641,
 0.874048806087641,
 0.8735240094463396,
 0.8737864077669902,
 0.871556022041459]
```

In [20]:
```
importances = pd.DataFrame({'feature':X_train.columns,'importance':np.round(clf.feature_importances_,3)})
importances = importances.sort_values('importance',ascending=False)
importances
```

Out[20]:

	feature	importance
0	age	0.973
1	Yhousing	0.014
3	Yloan	0.012
2	Ydefault	0.000

In [21]:
```
x3 = [[50, 1, 0, 0]]
pred2 = clf.predict(x3)
print("X=%s, Predicted=%s" % (x3[0], pred2[0]))
```

```
X=[50, 1, 0, 0], Predicted=0
```

Results

With an accuracy of 87%, this model is pretty effective at predicting the possibility of acceptance for a candidate if you know their age, whether they have a housing loan, whether they have a normal loan, and whether they have defaulted.

Of these factors, this model finds the candidate's age to be the most important, gives a little relevance to whether the candidate accepted a housing loan, and even less relevance to whether they have a normal loan. Whether they candidate defaulted or not has no relevance in the model and can be removed. This is probably because these variables are all highly correlated, so a vast majority of the information in the latter three variables are already given by a candidate's age.

So if a candidate is 50 years old and has a housing loan but not a normal loan and never defaulted, they are not expected to accept the bank's offer.

Final Logistical Regression Testing Candidate Characteristics and Likelihood of Accepting Offer

In [22]:
```
X= X.drop(['Ydefault'], axis=1)
X= sm.add_constant(X)
y= df['accepted']
logit1 = sm.Logit(y,X)
results = logit1.fit()
results.summary()
```

```
Optimization terminated successfully.
         Current function value: 432.821231
         Iterations 6

C:\Users\Woyte\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning: Method .ptp is
deprecated and will be removed in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
C:\Users\Woyte\Anaconda3\lib\site-packages\statsmodels\base\model.py:492: HessianInversionWarning: Invert
ing hessian failed, no bse or cov_params available
  'available', HessianInversionWarning)
C:\Users\Woyte\Anaconda3\lib\site-packages\statsmodels\base\model.py:492: HessianInversionWarning: Invert
ing hessian failed, no bse or cov_params available
  'available', HessianInversionWarning)
C:\Users\Woyte\Anaconda3\lib\site-packages\statsmodels\discrete\discrete_model.py:3390: RuntimeWarning: d
ivide by zero encountered in double_scalars
  return 1 - self.llf/self.llnull
```

Out[22]:

Logit Regression Results

Dep. Variable:	accepted	**No. Observations:**	30488
Model:	Logit	**Df Residuals:**	30484
Method:	MLE	**Df Model:**	3
Date:	Thu, 17 Oct 2019	**Pseudo R-squ.:**	inf
Time:	02:15:41	**Log-Likelihood:**	-1.3196e+07
converged:	True	**LL-Null:**	0.0000
Covariance Type:	nonrobust	**LLR p-value:**	1.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-2.4922	0.069	-36.029	0.000	-2.628	-2.357
age	0.0135	0.002	8.471	0.000	0.010	0.017
Yhousing	0.0610	0.035	1.757	0.079	-0.007	0.129
Yloan	-0.0418	0.048	-0.869	0.385	-0.136	0.052

In [23]:
```
X= df[['age'] +['Yhousing']]
X=sm.add_constant(X)
logit1 = sm.Logit(y,X)
results = logit1.fit()
results.summary()
```

```
C:\Users\Woyte\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2389: FutureWarning: Method .ptp is
deprecated and will be removed in a future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)

Optimization terminated successfully.
         Current function value: 432.795705
         Iterations 6

C:\Users\Woyte\Anaconda3\lib\site-packages\statsmodels\base\model.py:492: HessianInversionWarning: Invert
ing hessian failed, no bse or cov_params available
  'available', HessianInversionWarning)
C:\Users\Woyte\Anaconda3\lib\site-packages\statsmodels\base\model.py:492: HessianInversionWarning: Invert
ing hessian failed, no bse or cov_params available
  'available', HessianInversionWarning)
C:\Users\Woyte\Anaconda3\lib\site-packages\statsmodels\discrete\discrete_model.py:3390: RuntimeWarning: d
ivide by zero encountered in double_scalars
  return 1 - self.llf/self.llnull
```

Out[23]:

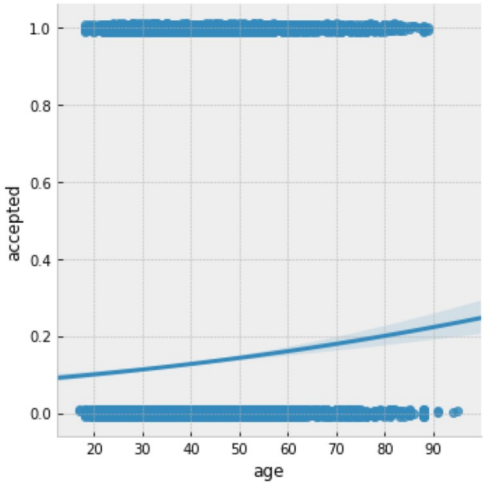
Logit Regression Results

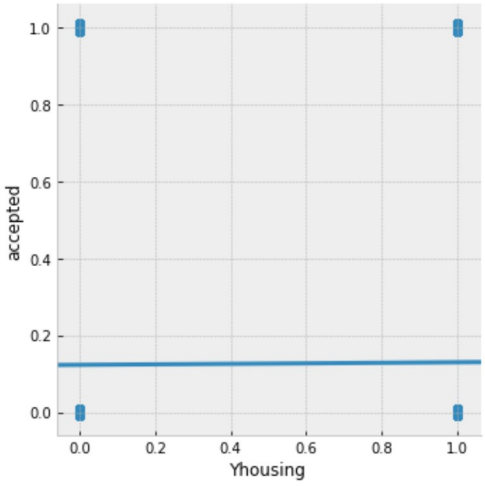
Dep. Variable:	accepted	**No. Observations:**	30488
Model:	Logit	**Df Residuals:**	30485
Method:	MLE	**Df Model:**	2
Date:	Thu, 17 Oct 2019	**Pseudo R-squ.:**	inf
Time:	02:15:41	**Log-Likelihood:**	-1.3195e+07
converged:	True	**LL-Null:**	0.0000
Covariance Type:	nonrobust	**LLR p-value:**	1.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-2.4984	0.069	-36.313	0.000	-2.633	-2.364
age	0.0135	0.002	8.481	0.000	0.010	0.017
Yhousing	0.0596	0.035	1.718	0.086	-0.008	0.128

In [24]:
```
plt.style.use('bmh')
sns.lmplot(x='age', y='accepted', data=df, logistic=True, y_jitter=.01)
sns.lmplot(x='Yhousing', y='accepted', data=df, logistic=True, y_jitter=.01)
```

Out[24]:
```
<seaborn.axisgrid.FacetGrid at 0x200dbd01c88>
```





In [25]:
```
np.exp(results.params)
```

Out[25]:
```
const       0.082214
age         1.013589
Yhousing    1.061400
dtype: float64
```

Results

Of all the candidate characteristics, only a candidate's age and if they they had a housing loan were statistically significant at a 90% confidence level, and at a 95% confidence level the housing loan variable was dropped. That meant that at 95% confidence and above, only a candidate's age had any signicant relationship with their chance of accepting a loan.

The model at a 90% confidence level is $Accepted = -2.5 + .0135(Age) + .035(Yhousing) + E$

Otherwise, the odds ratio shows that someone with a housing loan is 1.06 times as likely to accept the bank's offer than someone without, an interesting find, and every increase in someone's age by a single year increases their odds of accepting the bank's offer by 1.014 times as much.

In [ ]:
```
```