Libraries

In [1]:
```python
__author__ = "Isaac Taylor"
```

In [2]:
```python
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
import numpy as np
import seaborn as sns
import numpy as np
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE
from sklearn.linear_model import RidgeCV, LassoCV, Ridge, Lasso
import plotly.graph_objects as go
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import statsmodels.formula.api as smf
from statsmodels.tools.eval_measures import rmse
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
```

Loading Data

In [3]:
```python
import ssl
ssl._create_default_https_context = ssl._create_unverified_context
URL = 'https://drive.google.com/file/d/15Z_GFFqDdH0jnf3t7APB3_49cuofLK2v/view?usp=sharing'
path = 'https://drive.google.com/uc?export=download&id=' + URL.split('/')[-2]
pwd_data = pd.read_csv(path)
```

Feature Importance

In [4]:
```python
#previewing subset of data
pwd_data[999970:999975]
```

Out[4]:

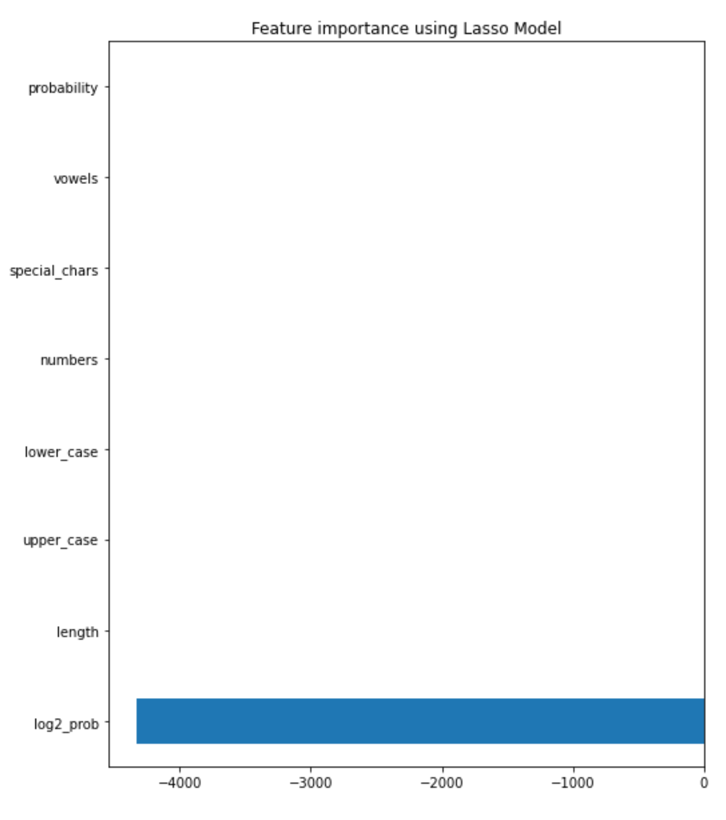
	password	length	upper_case	lower_case	numbers	special_chars	vowels	probability	log2_prob
999970	Nintendo123	11	1	7	3	0	3	2.936912e-13	-41.630765
999971	ninoska1212	11	0	7	4	0	3	2.936876e-13	-41.630783
999972	Mysterious	10	1	9	0	0	4	2.936671e-13	-41.630883
999973	143pragati	10	0	7	3	0	3	2.936389e-13	-41.631022
999974	keujdjq	7	0	7	0	0	2	2.936384e-13	-41.631024

In [5]:
```python
# adding number of a tries it will take to crack password
pwd_data['tries'] = [i+1 for i in range(len(pwd_data))]
```

In [6]:
```python
X = pwd_data.drop(['tries' ,'password'],1)   #Feature Matrix
y = pwd_data["tries"]          #Target Variable
reg = LassoCV(tol=.5)
reg.fit(X, y)
print("Best alpha using built-in LassoCV: %f" % reg.alpha_)
print("Best score using built-in LassoCV: %f" %reg.score(X,y))
coef = pd.Series(reg.coef_, index = X.columns)
print("Lasso picked " + str(sum(coef != 0)) + " variables and eliminated the other " +  str(sum(coef == 0)) + " variab
les")
imp_coef = coef.sort_values()
import matplotlib
matplotlib.rcParams['figure.figsize'] = (8.0, 10.0)
imp_coef.plot(kind = "barh")
plt.title("Feature importance using Lasso Model")
```

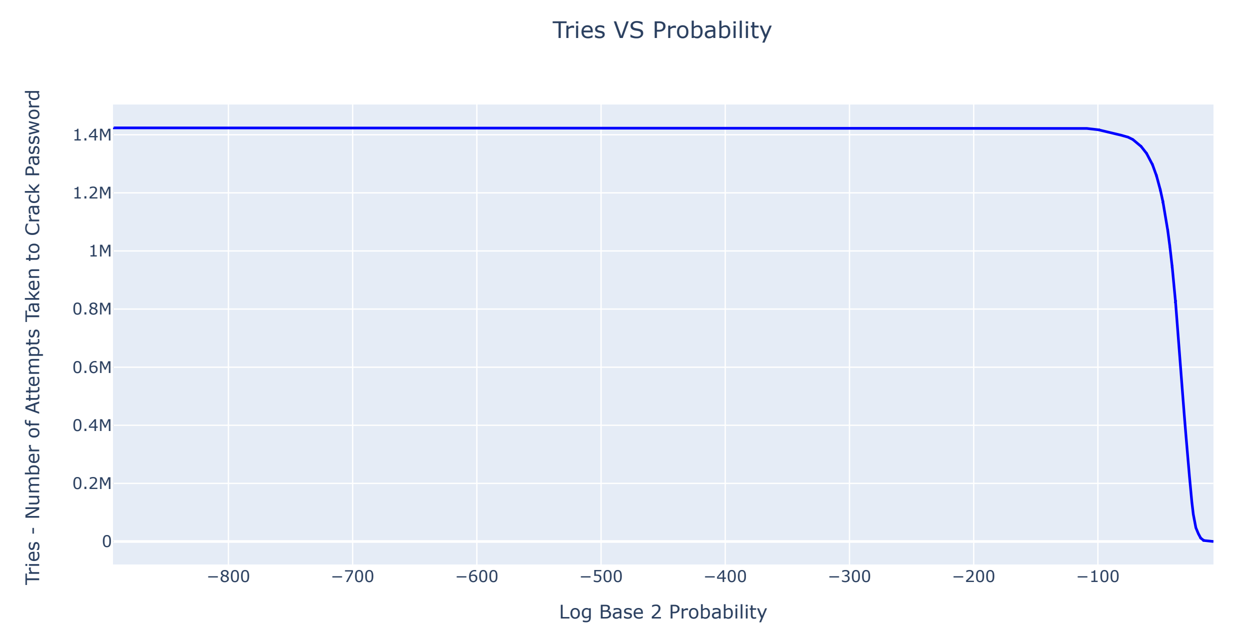
```
Best alpha using built-in LassoCV: 1959873.841155
Best score using built-in LassoCV: 0.192967
Lasso picked 2 variables and eliminated the other 6 variables
```

Out[6]:
```
Text(0.5, 1.0, 'Feature importance using Lasso Model')
```



Visualizing Data

In [7]:
```python
# plotly figure
fig = go.Figure()

fig.add_trace(go.Scatter(
    x=pwd_data['log2_prob'],
    y=pwd_data['tries'],
    marker=dict(
        color="blue"
    ),
    showlegend=False
))

fig.update_layout(
    title="Tries VS Probability", title_x=0.5,
    xaxis_title="Log Base 2 Probability",
    yaxis_title="Tries - Number of Attempts Taken to Crack Password"
)


fig.show()
```



Classifying Passwords - Likely vs Unlikely to be the Actual Password

In [8]:
```python
#based on Tries VS Probability graph
SEPARATOR = -120.0 #based on Tries VS Probability graph
pwd_data['LIKELY'] = [1 if x > SEPARATOR else 0 for x in pwd_data['log2_prob']]
#preview
pwd_data.head()
```

Out[8]:

	password	length	upper_case	lower_case	numbers	special_chars	vowels	probability	log2_prob	tries	LIKELY
0	123	3	0	0	3	0	0	0.008033	-6.959867	1	1
1	200	3	0	0	3	0	0	0.003074	-8.345746	2	1
2	234	3	0	0	3	0	0	0.002709	-8.528232	3	1
3	198	3	0	0	3	0	0	0.002560	-8.609877	4	1
4	199	3	0	0	3	0	0	0.002234	-8.806298	5	1

Decision Tree Classification

In [9]:
```python
x = pwd_data.drop(['password','LIKELY'], axis=1)
y = pwd_data['LIKELY']
#train, test, split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = .30, random_state = 42)
dt_model = DecisionTreeClassifier()
dt_model.fit(x_train, y_train)
#prediction
y_pred = dt_model.predict(x_test)
```

In [10]:
```python
#accuracy
accuracy = metrics.accuracy_score(y_test, y_pred)
print(accuracy)
```

```
0.9999976606819112
```

Logistic Regression Classification

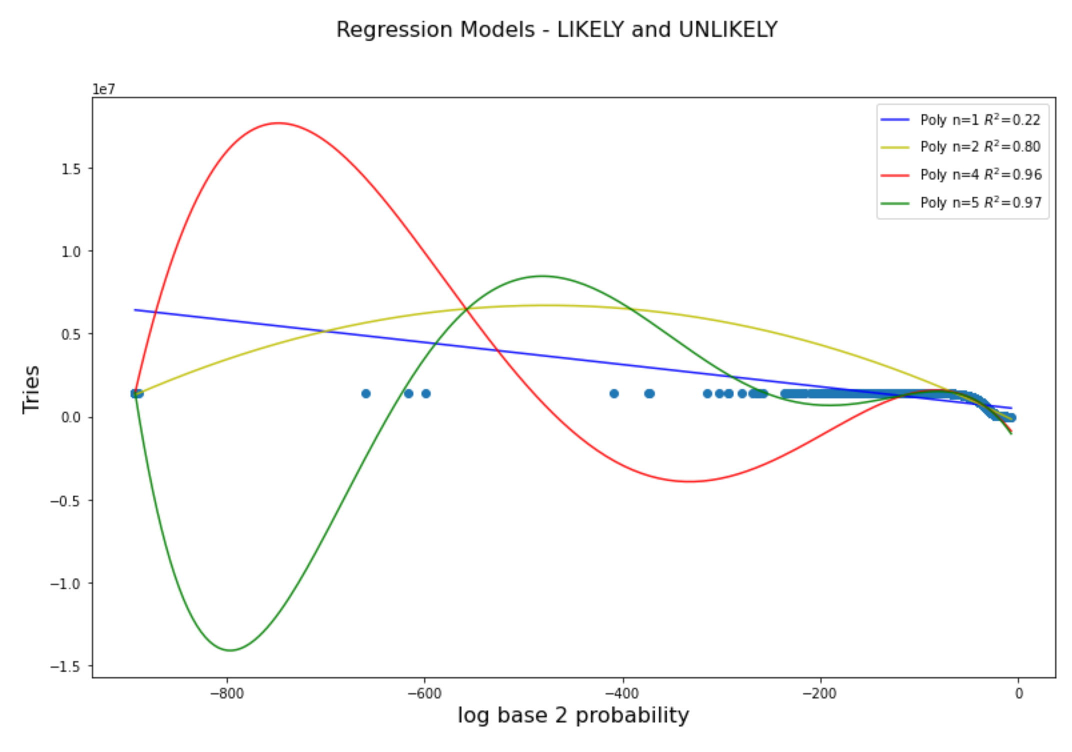
In [11]:
```python
#train, test, split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = .30, random_state = 42)
lg_model = LogisticRegression(solver='lbfgs', max_iter=400)
lg_model.fit(x_train, y_train)
#prediction
y_pred = lg_model.predict(x_test)
#accuracy
accuracy = metrics.accuracy_score(y_test, y_pred)
print(accuracy)
```

```
0.9975366980525177
```

Predicting Number of Tries with Linear Regression

In [12]:
```python
fig = plt.figure(figsize=(8 * 1.618, 8))

plt.scatter(pwd_data['log2_prob'], pwd_data['tries'])

df1 = pwd_data

poly_1 = smf.ols(formula='tries ~ 1 + log2_prob', data=df1).fit()
x = pd.DataFrame({'log2_prob': np.linspace(df1.log2_prob.min(), df1.log2_prob.max(), 200)})
plt.plot(x.log2_prob, poly_1.predict(x), 'b-', label='Poly n=1 $R^2$=%.2f' % poly_1.rsquared, alpha=0.9)

# 2-rd order polynomial
poly_2 = smf.ols(formula='tries ~ 1 + log2_prob + I(log2_prob ** 2.0)', data=df1).fit()
plt.plot(x.log2_prob, poly_2.predict(x), 'y-', alpha=0.9, label='Poly n=2 $R^2$=%.2f' % poly_2.rsquared)

# 4-rd order polynomial
poly_4 = smf.ols(formula='tries ~ 1 + log2_prob + I(log2_prob ** 2.0) + I(log2_prob ** 3.0) + I(log2_prob ** 4.0)', da
ta=df1).fit()
plt.plot(x.log2_prob, poly_4.predict(x), 'r-', alpha=0.9, label='Poly n=4 $R^2$=%.2f' % poly_4.rsquared)

# 5-rd order polynomial
poly_5 = smf.ols(formula='tries ~ 1 + log2_prob + I(log2_prob ** 2.0) + I(log2_prob ** 3.0) + I(log2_prob ** 4.0) + I(
log2_prob ** 5.0)', data=df1).fit()
plt.plot(x.log2_prob, poly_5.predict(x), 'g-', alpha=0.9, label='Poly n=5 $R^2$=%.2f' % poly_5.rsquared)

fig.suptitle('Regression Models - LIKELY and UNLIKELY', fontsize=16)
plt.xlabel('log base 2 probability', fontsize=16)
plt.ylabel('Tries', fontsize=16)

plt.legend()
```

Out[12]:
```
<matplotlib.legend.Legend at 0x7ffe3589ef90>
```



In [13]:
```python
#going with the fith degree polynomial model
#RMSE
X = pwd_data['log2_prob']
y = pwd_data['tries']
ypre = poly_5.predict(X)
# calc rmse
rmse5 = rmse(y, ypre)
print(poly_5.params)
print('\n rmse is ', rmse5)
```

```
Intercept              -1.692525e+06
log2_prob              -9.989258e+04
I(log2_prob ** 2.0)    -1.044435e+03
I(log2_prob ** 3.0)    -4.192885e+00
I(log2_prob ** 4.0)    -6.534137e-03
I(log2_prob ** 5.0)    -3.374506e-06
dtype: float64

 rmse is  67902.72244939854
```

Additional Analysis - Dropping Unlikely Passwords

In [14]:
```python
print('Before ',len(pwd_data))
#LIKELY only
likely = pwd_data[pwd_data['LIKELY'] == 1]
print('After ', len(likely))
#preview
likely.tail()
```

```
Before  1424915
After  1422492
```

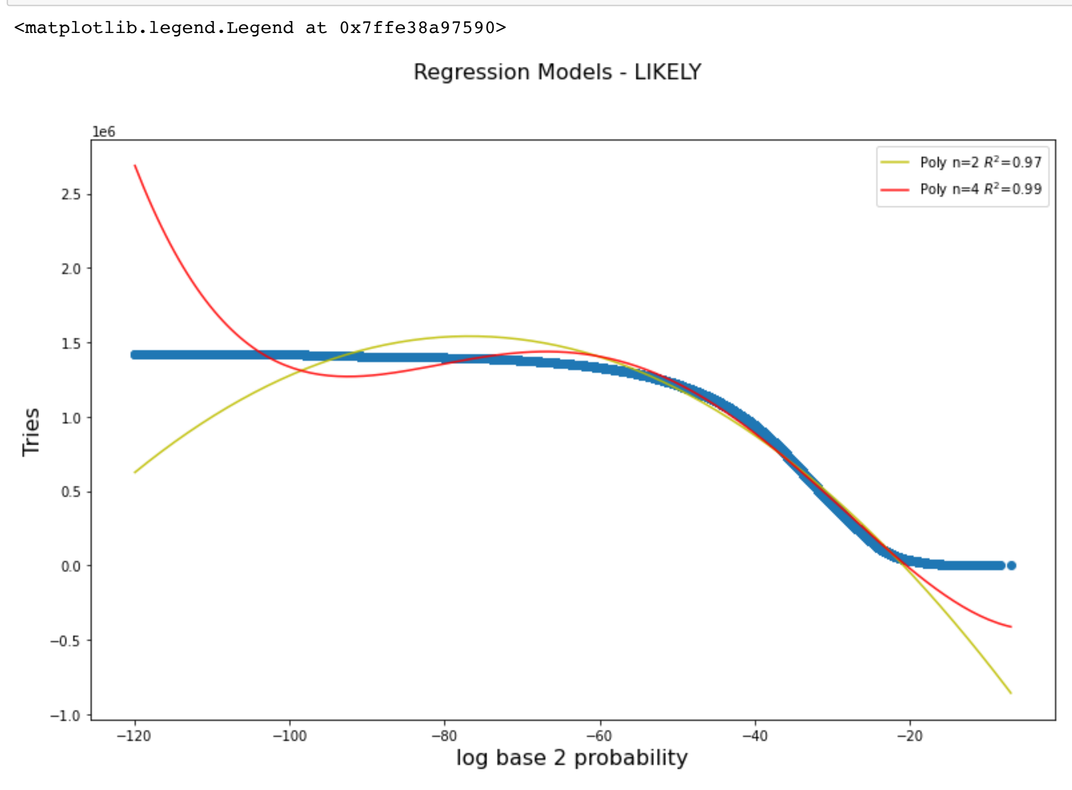
Out[14]:

	password	length	upper_case	lower_case	numbers	special_chars	vowels	probability	log2_prob	tries	LIKELY
1422487	337s76/37#But@5*6.	20	1	3	9	7	1	8.260361e-37	-119.865135	1422488	1
1422488	biohazard63119119biohazard	26	0	18	8	0	8	8.197056e-37	-119.876234	1422489	1
1422489	0511322765066006681e50n80	25	0	2	23	0	1	7.963195e-37	-119.917992	1422490	1
1422490	myazka_furqan@yahoo.co.id	25	0	21	0	4	9	7.901996e-37	-119.929122	1422491	1
1422491	erdemsahinkiymik5353032663	26	0	16	10	0	6	7.859210e-37	-119.936955	1422492	1

Linear Regression on Likely

In [15]:
```python
fig = plt.figure(figsize=(8 * 1.618, 8))

plt.scatter(likely['log2_prob'], likely['tries'])
df1 = likely
x = pd.DataFrame({'log2_prob': np.linspace(df1.log2_prob.min(), df1.log2_prob.max(), 200)})

# 2-rd order polynomial
poly_2 = smf.ols(formula='tries ~ 1 + log2_prob + I(log2_prob ** 2.0)', data=df1).fit()
plt.plot(x.log2_prob, poly_2.predict(x), 'y-', alpha=0.9, label='Poly n=2 $R^2$=%.2f' % poly_2.rsquared)

# 4-rd order polynomial
poly_4 = smf.ols(formula='tries ~ 1 + log2_prob + I(log2_prob ** 2.0) + I(log2_prob ** 3.0) + I(log2_prob ** 4.0)', da
ta=df1).fit()
plt.plot(x.log2_prob, poly_4.predict(x), 'r-', alpha=0.9, label='Poly n=4 $R^2$=%.2f' % poly_4.rsquared)

fig.suptitle('Regression Models - LIKELY', fontsize=16)
plt.xlabel('log base 2 probability', fontsize=16)
plt.ylabel('Tries', fontsize=16)
plt.legend()
```

Out[15]:
```
<matplotlib.legend.Legend at 0x7ffe38a97590>
```



In [16]:
```python
X = df1['log2_prob']
y = df1['tries']
ypred = poly_2.predict(X)
# calc rmse
rmse1 = rmse(y, ypred)
print(poly_2.params)
print('\n rmse is ', rmse1)
```

```
Intercept              -1.361021e+06
log2_prob              -7.556119e+04
I(log2_prob ** 2.0)    -4.919591e+02
dtype: float64

 rmse is  66768.30079278407
```

In [17]:
```python
# calc rmse
ypred = poly_4.predict(X)
rmse2 = rmse(y, ypred)
print(poly_4.params)
print('\n rmse is ', rmse2)
```

```
Intercept              -391522.073409
log2_prob                20015.215713
I(log2_prob ** 2.0)       2713.500203
I(log2_prob ** 3.0)         43.268947
I(log2_prob ** 4.0)          0.198609
dtype: float64

 rmse is  47797.23201722325
```

Conclusion:

The polynomial model of degree 4 seems to fit the Likely password susbset best.

In [ ]:
```python

```