Predicting credit card fraud

In [133]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

In [135]:
```
df = pd.read_csv("/Users/snehilshandilya/Desktop/fraudTest_2.csv")
```

In [136]:
```
df.head()
```

Out[136]:

	Unnamed: 0	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	...	la
0	0	2020-06-21 12:14:25	2291163933867244	fraud_Kirlin and Sons	personal_care	2.86	Jeff	Elliott	M	351 Darlene Green	...	33.965
1	1	2020-06-21 12:14:33	3573030041201292	fraud_Sporer-Keebler	personal_care	29.84	Joanne	Williams	F	3638 Marsh Union	...	40.320
2	2	2020-06-21 12:14:53	3598215285024754	fraud_Swaniawski, Nitzsche and Welch	health_fitness	41.28	Ashley	Lopez	F	9333 Valentine Point	...	40.672
3	3	2020-06-21 12:15:15	3591919803438423	fraud_Haley Group	misc_pos	60.05	Brian	Williams	M	32941 Krystal Mill Apt. 552	...	28.569
4	4	2020-06-21 12:15:17	3526826139003047	fraud_Johnston-Casper	travel	3.19	Nathan	Massey	M	5783 Evan Roads Apt. 465	...	44.252

5 rows × 23 columns

In [137]:
```
df.columns
```

Out[137]:
```
Index(['Unnamed: 0', 'trans_date_trans_time', 'cc_num', 'merchant', 'category',
       'amt', 'first', 'last', 'gender', 'street', 'city', 'state', 'zip',
       'lat', 'long', 'city_pop', 'job', 'dob', 'trans_num', 'unix_time',
       'merch_lat', 'merch_long', 'is_fraud'],
      dtype='object')
```

2.) Selecting four columns to use as features

In [138]:
```
df_select = df[["trans_date_trans_time", "category", "amt", "city_pop", "is_fraud"]]
```

In [139]:
```
df_select.head()
```

Out[139]:

	trans_date_trans_time	category	amt	city_pop	is_fraud
0	2020-06-21 12:14:25	personal_care	2.86	333497	0
1	2020-06-21 12:14:33	personal_care	29.84	302	0
2	2020-06-21 12:14:53	health_fitness	41.28	34496	0
3	2020-06-21 12:15:15	misc_pos	60.05	54767	0
4	2020-06-21 12:15:17	travel	3.19	1126	0

In [ ]:

In [ ]:

3.) Creating a unique variable out of trans_date. (Here, variable used: hour)

In [140]:
```
df["trans_date_trans_time"] = pd.to_datetime(df["trans_date_trans_time"])
```

In [141]:
```
df_select["time_var"] = [i.hour for i in df["trans_date_trans_time"]]
```

```
/var/folders/sv/s309_3dd79s_59j12prhgcd00000gn/T/ipykernel_46023/1545706583.py:1: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.
html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
tml#returning-a-view-versus-a-copy)
  df_select["time_var"] = [i.hour for i in df["trans_date_trans_time"]]
```

In [142]:
```
df_select["time_var"]
```

Out[142]:
```
0         12
1         12
2         12
3         12
4         12
          ..
555714    23
555715    23
555716    23
555717    23
555718    23
Name: time_var, Length: 555719, dtype: int64
```

In [143]:
```
X = pd.get_dummies(df_select["category"])
Y = df_select["is_fraud"]
```

In [144]:
```
dummies = pd.get_dummies(df_select["category"])
X = pd.concat([dummies, df_select[["amt", "city_pop", "time_var"]]], axis = 1)
Y = df_select["is_fraud"]
```

In [145]:
```
X.head()
```

Out[145]:

	entertainment	food_dining	gas_transport	grocery_net	grocery_pos	health_fitness	home	kids_pets	misc_net	misc_pos	personal_care
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	1	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	1	0
4	0	0	0	0	0	0	0	0	0	0	0

5.) Training a Logistic regression.

In [178]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_normalized = scaler.fit_transform(X)
```

In [179]:
```
from sklearn.linear_model import LogisticRegression
```

In [180]:
```
log_reg = LogisticRegression().fit(X_normalized,Y)
```

In [ ]:
```

```

6.) Setting a False Positive threshold of 5%

In [181]:
```
from sklearn.metrics import confusion_matrix
import numpy as np
```

In [187]:
```
from sklearn.metrics import confusion_matrix

# Fit your Logistic Regression model
log_reg.fit(X, Y)

# Predict on your test set
y_pred = log_reg.predict_proba(X)[:, 1]

# Find the threshold that gives you the desired False Negative percentage
target_fnr = 0.05
fnr = 1
threshold = 0
while fnr > target_fnr:
    threshold += 0.001
    y_pred_threshold = y_pred > threshold
    fnr = confusion_matrix(Y, y_pred_threshold)[1,0] / (confusion_matrix(Y, y_pred_threshold)[1,0] + conf

# Use the threshold to predict on your test set
y_pred_threshold = y_pred > threshold
```

In [188]:
```
confusion_matrix = confusion_matrix(Y, y_pred_threshold)
print(confusion_matrix)
```

```
[[  2397 551177]
 [     5   2140]]
```

In [189]:
```
y_pred_threshold
```

Out[189]:
```
array([ True,  True,  True, ...,  True,  True,  True])
```

7.) Calculating profits if the company makes .02*amt on True transactions and loses -amt on False

In [190]:
```
#sum(x1 * TP + x2*FP +x3*FN + x4*TN) two of these Xs will be 0
#Values only declared for two values

df_temp = df_select.copy()
df_temp["pred"] = log_reg.predict(X)
df_temp = df_temp[["pred", "is_fraud", "amt"]]
```

In [191]:
```
df_temp.head()
```

Out[191]:

	pred	is_fraud	amt
0	0	0	2.86
1	0	0	29.84
2	0	0	41.28
3	0	0	60.05
4	0	0	3.19

In [192]:
```
import numpy as np

# Multiply the true transactions by 0.2
confusion_matrix[0,0] = confusion_matrix[0,0] * 0.2
confusion_matrix[1,1] = confusion_matrix[1,1] * 0.2

# Multiply the false transactions by -1
confusion_matrix[0,1] = confusion_matrix[0,1] * -1
confusion_matrix[1,0] = confusion_matrix[1,0] * -1

print(confusion_matrix)

((confusion_matrix[0,0] * 0.2) + (confusion_matrix[1,1] * 0.2)) - ((confusion_matrix[0,1] * -1) + (confu
```

```
[[    479 -551177]
 [     -5     428]]
```

Out[192]:
```
-551000.6
```

8.) Using Logistic Regression Lasso to inform which variables are trusted

In [193]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import make_classification

# Initialize the logistic Lasso regression model
log_reg1 = LogisticRegression(penalty="l1", solver='liblinear')

# Fit the model to the data
log_reg1.fit(X,Y)

# Predict using the model
y_pred = log_reg1.predict(X)
```

In [194]:
```
coefficients = log_reg1.coef_
coefficients
```

Out[194]:
```
array([[-4.95110329e+00, -5.02084956e+00, -3.03754808e+00,
        -3.23165867e+00, -1.97104465e+00, -5.08661261e+00,
        -5.20467713e+00, -5.15680488e+00, -2.39303075e+00,
        -4.23311600e+00, -4.86125140e+00, -2.90640168e+00,
        -4.07139257e+00, -9.27893554e+00,  1.99996936e-03,
        -5.05069785e-07,  1.19726164e-01]])
```

In [195]:
```
df2 = pd.DataFrame(zip(X.columns, coefficients))
df2
```

Out[195]:

	0	1
0	entertainment	[-4.951103289444473, -5.0208495637382695, -3.0...

In [196]:
```
X.columns.shape
```

Out[196]:
```
(17,)
```

In [197]:
```
coefs = (X.columns)
```

In [198]:
```
coefs
```

Out[198]:
```
Index(['entertainment', 'food_dining', 'gas_transport', 'grocery_net',
       'grocery_pos', 'health_fitness', 'home', 'kids_pets', 'misc_net',
       'misc_pos', 'personal_care', 'shopping_net', 'shopping_pos', 'travel',
       'amt', 'city_pop', 'time_var'],
      dtype='object')
```

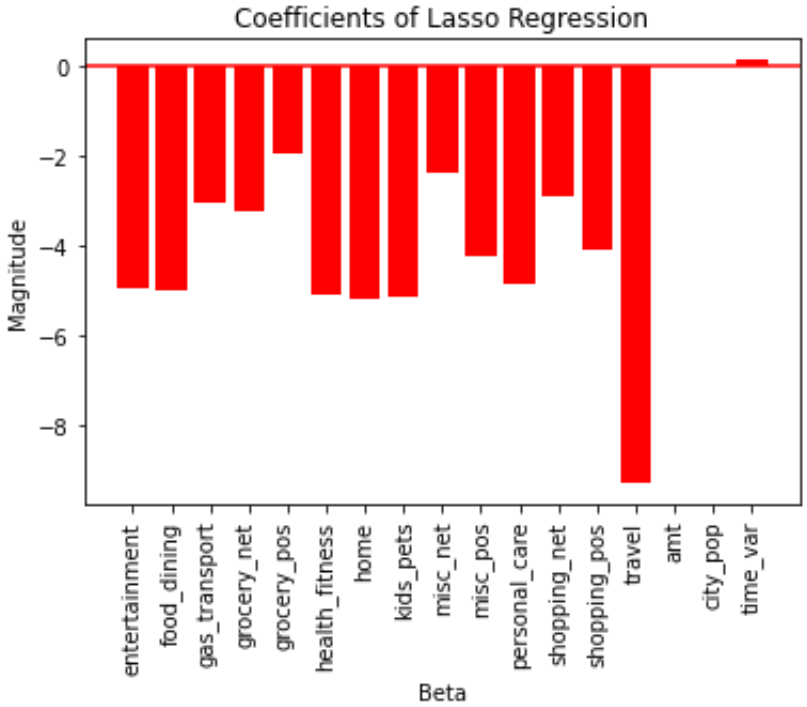
In [199]:
```
coefficients[0]
```

Out[199]:
```
array([-4.95110329e+00, -5.02084956e+00, -3.03754808e+00, -3.23165867e+00,
       -1.97104465e+00, -5.08661261e+00, -5.20467713e+00, -5.15680488e+00,
       -2.39303075e+00, -4.23311600e+00, -4.86125140e+00, -2.90640168e+00,
       -4.07139257e+00, -9.27893554e+00,  1.99996936e-03, -5.05069785e-07,
        1.19726164e-01])
```

In [200]:
```
import matplotlib.pyplot as plt
```

In [201]:
```
plt.bar(coefs, coefficients[0], color = "red")
plt.axhline(0, color = "red")
plt.xticks(np.arange(len(coefs)), coefs, rotation='vertical')
plt.xlabel("Beta")
plt.ylabel("Magnitude")
plt.title("Coefficients of Lasso Regression")
```

Out[201]:
```
Text(0.5, 1.0, 'Coefficients of Lasso Regression')
```



The variables which are close to 0 are 'amt' and 'city_pop', signifying that these variables are not helpful.

In [ ]: