# 1.) Import the Credit Card Fraud Data From CCLE

```
import pandas as pd
from google.colab import drive
import matplotlib.pyplot as plt
import numpy as np
```

```
drive.mount('/content/gdrive/', force_remount = True)
```

```
Mounted at /content/gdrive/
```

```
df = pd.read_csv("/content/gdrive/MyDrive/Econ441B/fraudTest.csv")
```

```
df.head()
```

| | Unnamed: 0 | trans_date_trans_time | cc_num | merchant | categor |
|---|---|---|---|---|---|
| **0** | 0 | 2020-06-21 12:14:25 | 2291163933867244 | fraud_Kirlin and Sons | personal_ca |
| **1** | 1 | 2020-06-21 12:14:33 | 3573030041201292 | fraud_Sporer-Keebler | personal_ca |
| **2** | 2 | 2020-06-21 12:14:53 | 3598215285024754 | fraud_Swaniawski, Nitzsche and Welch | health_fitnes |
| **3** | 3 | 2020-06-21 12:15:15 | 3591919803438423 | fraud_Haley Group | misc_pc |
| **4** | 4 | 2020-06-21 12:15:17 | 3526826139003047 | fraud_Johnston-Casper | trav |

5 rows × 23 columns

# 2.) Select four columns to use as features (one just be trans_date_trans)

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

```
df_select.columns
```

```
Index(['trans_date_trans_time', 'category', 'amt', 'city_pop', 'is_fraud'],
dtype='object')
```

# 3.) Create a your own variable out of trans_date. Create dummies for factor vars

```
type(df_select["trans_date_trans_time"][0])
```

```
str
```

```
df_select["trans_date_trans_time"] = pd.to_datetime(df_select["trans_date_trans_time"])
```

```
<ipython-input-18-99f721e4ce0f>:1: SettingWithCopyWarning:
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/us
  df_select["trans_date_trans_time"] = pd.to_datetime(df_select["trans_date_trans_tim
```

```
dir(df_select["trans_date_trans_time"][0])
```

```
 'freq',
 'freqstr',
 'fromisocalendar',
 'fromisoformat',
 'fromordinal',
 'fromtimestamp',
 'hour',
 'is_leap_year',
 'is_month_end',
 'is_month_start',
 'is_quarter_end',
 'is_quarter_start',
 'is_year_end',
 'is_year_start',
 'isocalendar',
 'isoformat',
 'isoweekday',
 'max',
```

```
 'microsecond',
 'min',
 'minute',
 'month',
 'month_name',
 'nanosecond',
 'normalize',
 'now',
 'quarter',
 'replace',
 'resolution',
 'round',
 'second',
 'strftime',
 'strptime',
 'time',
 'timestamp',
 'timetuple',
 'timetz',
 'to_datetime64',
 'to_julian_date',
 'to_numpy',
 'to_period',
 'to_pydatetime',
 'today',
 'toordinal',
 'tz',
 'tz_convert',
 'tz_localize',
 'tzinfo',
 'tzname',
 'utcfromtimestamp',
 'utcnow',
 'utcoffset',
 'utctimetuple',
 'value',
 'week',
 'weekday',
 'weekofyear',
 'year']
```

```
df_select["time_var"] = [i.second for i in df_select["trans_date_trans_time"]]
```

```
<ipython-input-20-fa4370ef92e9>:1: SettingWithCopyWarning:
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/us
  df_select["time_var"] = [i.second for i in df_select["trans_date_trans_time"]]
```

```
X = pd.get_dummies(df_select, ["category"]).drop(["trans_date_trans_time", "is_fraud"], ax
y = df["is_fraud"]
```

```
X.head()
```

| | amt | city_pop | time_var | category_entertainment | category_food_dining | catego |
|---|---|---|---|---|---|---|
| **0** | 2.86 | 333497 | 25 | 0 | 0 | |
| **1** | 29.84 | 302 | 33 | 0 | 0 | |
| **2** | 41.28 | 34496 | 53 | 0 | 0 | |
| **3** | 60.05 | 54767 | 15 | 0 | 0 | |
| **4** | 3.19 | 1126 | 17 | 0 | 0 | |

# XXX SKIP THIS WE WILL TALK ABOUT NEXT CLASS

```
resample_X = X
resample_y = y
```

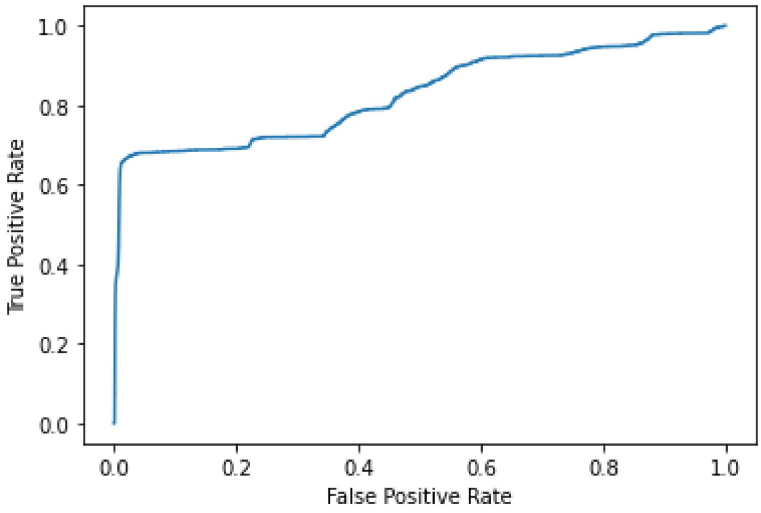
# 5.) Train a Logistic regression.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_normalized = scaler.fit_transform(resample_X)

from sklearn.linear_model import LogisticRegression



log_reg = LogisticRegression().fit(X_normalized, resample_y)

y_pred = log_reg.predict(X_normalized)
```

# 6.) The company you are working for wants to target at a False Positive rate of 5% what threshold should you use? (Use oversampled data)

```
# ASK chatgpt
# If you dont like the code ask in a different way

from sklearn.metrics import roc_auc_score, roc_curve
# Make predictions using the logistic regression model
```

```
y_pred_proba = log_reg.predict_proba(X_normalized)

# Calculate the ROC curve
fpr, tpr, thresholds = roc_curve(resample_y, y_pred_proba[:, 1])

# Plot the ROC curve
plt.plot(fpr, tpr)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')

# Find the threshold that corresponds to the desired FPR of 5%
index = np.where(fpr <= 0.05)[0]
threshold = thresholds[index[0]]

# Use this threshold for making predictions in the future
y_pred = (y_pred_proba[:,1] > threshold).astype(int)
```




```
print (threshold)
```

```
1.9999999999999998
```

# 7.) If the company makes .02*amt on True transactions and loses -amt on False (Use original data)

```
df_temp = df_select.copy()
```

```
df_temp["pred"] = log_reg.predict(resample_X)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/base.py:443: UserWarning: X has featur
  warnings.warn(
```

```
df_temp = df_temp[["pred", "is_fraud", "amt"]]
```

```
df_temp.head()
# Which combinations of "pred" and "if_fraud" do we profit .02*amt, which cases do
# we lose -amt??
```

| | 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 |

```
true_a = df_temp.loc[(df_temp['pred'] == 0) & (df_temp['is_fraud'] == 0), 'amt'].sum()*0.0
true_b = df_temp.loc[(df_temp['pred'] == 1) & (df_temp['is_fraud'] == 1), 'amt'].sum()*0.0
false_a = df_temp.loc[(df_temp['pred'] == 1) & (df_temp['is_fraud'] == 0), 'amt'].sum()
false_b = df_temp.loc[(df_temp['pred'] == 0) & (df_temp['is_fraud'] == 1), 'amt'].sum()
profit = true_a + true_b - false_a - false_b
profit
```

```
-5495046.260000001
```

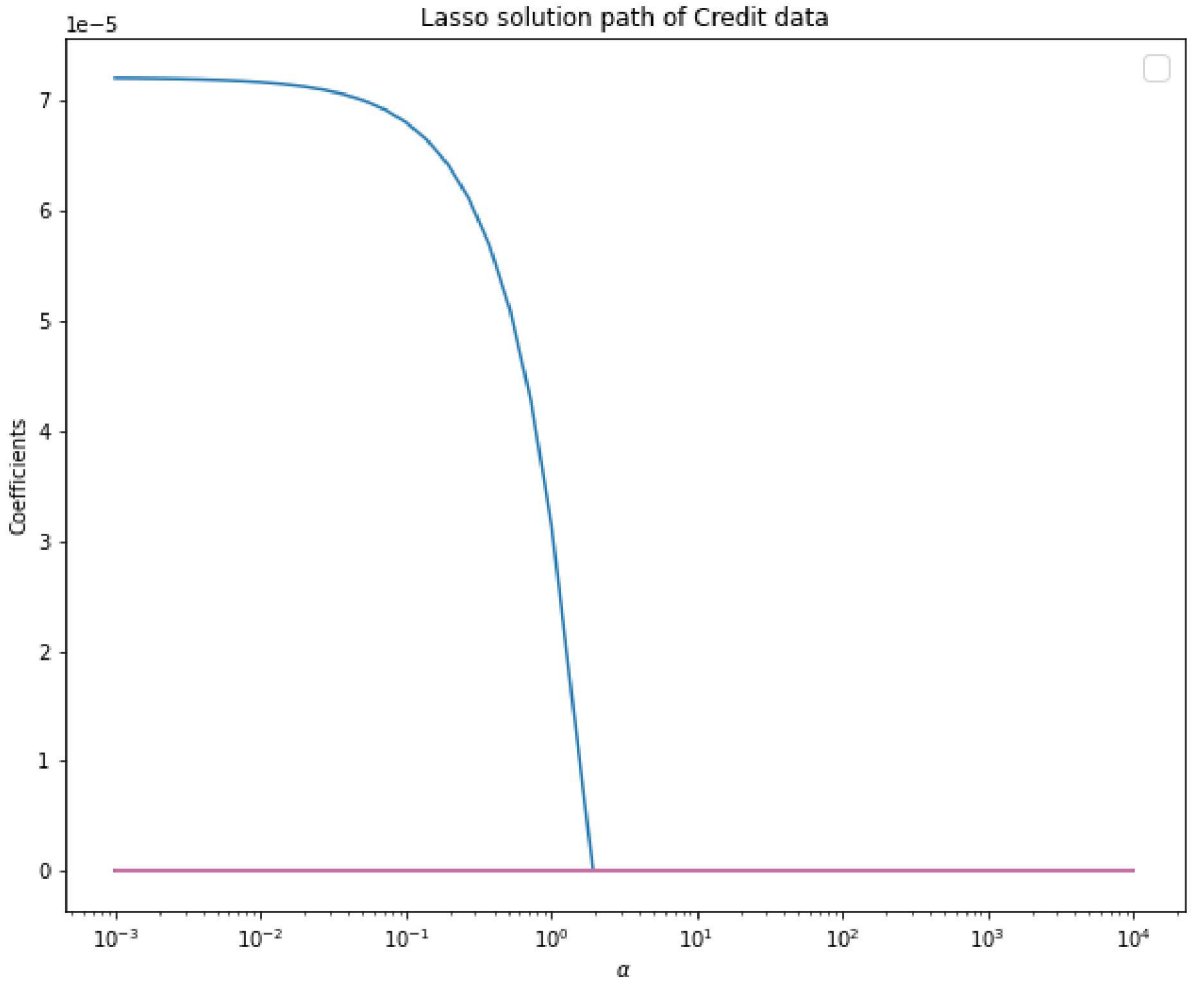
It can be seen that the model needs further adjustment to reduce the errors. This model will lead the company to lose big amount of money becasue of the fales positives and negatives.

# 8.) Using Logistic Regression Lasso to inform you. Would you use the selected features in a trusted prediction model?

```
# If most or all your variables go to 0 => Your data is garbage
# The regularization will tell us if our model has significance
# This of using coefficient strength similar to r^2


from sklearn.linear_model import Lasso

clf = Lasso()
# Ridge regularization parameter
alphas = np.logspace(start = -3, stop = 4, base = 10)
# Train the model with different regularization strengths
coefs = []
for a in alphas:
  clf.set_params(alpha = a)
  clf.fit(resample_X, resample_y)
```

```
  coefs.append(clf.coef_)


# Visualize Ridge solution path
plt.figure(figsize = (10, 8))
ax = plt.gca()
ax.plot(alphas, coefs)
ax.set_xscale("log")
ax.legend(fontsize = 16)
plt.xlabel(r"$\alpha$")
plt.ylabel("Coefficients")
plt.title("Lasso solution path of Credit data")
plt.axis("tight")
```

```
WARNING:matplotlib.legend:No handles with labels found to put in legend.
(0.00044668359215096305,
 22387.21138568338,
 -3.604626722271198e-06,
 7.567258194453309e-05)
```




it can be seen that all variables except one are approching to zero which requires to be eleminated or replaced.

Colab paid products - Cancel contracts here

✓ 0s completed at 21:19