INTRODUCTION TO PROBLEM AND DATASET

https://www.kaggle.com/datasets/lorenzozoppelletto/financial-risk-for-loan-approval?resource=download

Financial Risk, which involves the possibility of losing money, is crucial to businesses, corporate banking, and overall finance. Managing your financial risk can help an individual or organization maintain financial stability, make informed decisions about investments, and plan for the future.

In this project, I will use Linear Regression to predict the risk score of individuals based on their personal and financial data. I will be using the same dataset from my last project. This dataset from Kaggle, titled "Financial Risk For Loan Approval" by Lorenzo Zoppelltto contains 36 features and 20000 rows.

LINEAR REGRESSION

Linear regression is a supervised machine-learning model that is used to predict a continuous value. Linear regression can help us understand the relationship between a dependent variable and an independent variable. This can be done by finding the line of best fit also known as the regression line. The line of best fit is a linear line that minimizes the between the data points and itself.

Line of best-fit Equation

$y = mx + b$

y is the predicted value

x is the feature/independent variable

m is the slope of the line

b is the y-intercept

This shows the equation of a regression line for one independent variable if you wanted to add more variables it would look somethong like this.

$y = mx_1 + mx_2 + mx_3 + mx_4 + \ldots + b$

During this project, I will evaluate my model by using mean squared error, r^2, and root mean squared error.

Mean squared error (mse) finds the average squared differences between the predicted values and the actual values.

$$\text{MSE} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}}_{\text{Mean}}\underbrace{(Y_i - \hat{Y}_i)}_{\text{Error}}\underbrace{^2}_{\text{Squared}}$$

n = total number of data poinyes

y_i = actual value

$\hat{y}$ = predicted value

R-squared also known as the coefficient of determination measures the variance of the model. This number is between -1 and 1, the closer it is to 1 the better the model is.

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \mu)^2}.$$

y_i = actual value

$\hat{y}$ = predicted value

μ = mean of Y_i

Root mean squared error (rmse) measures the difference between a model's predicted values and the actual values.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

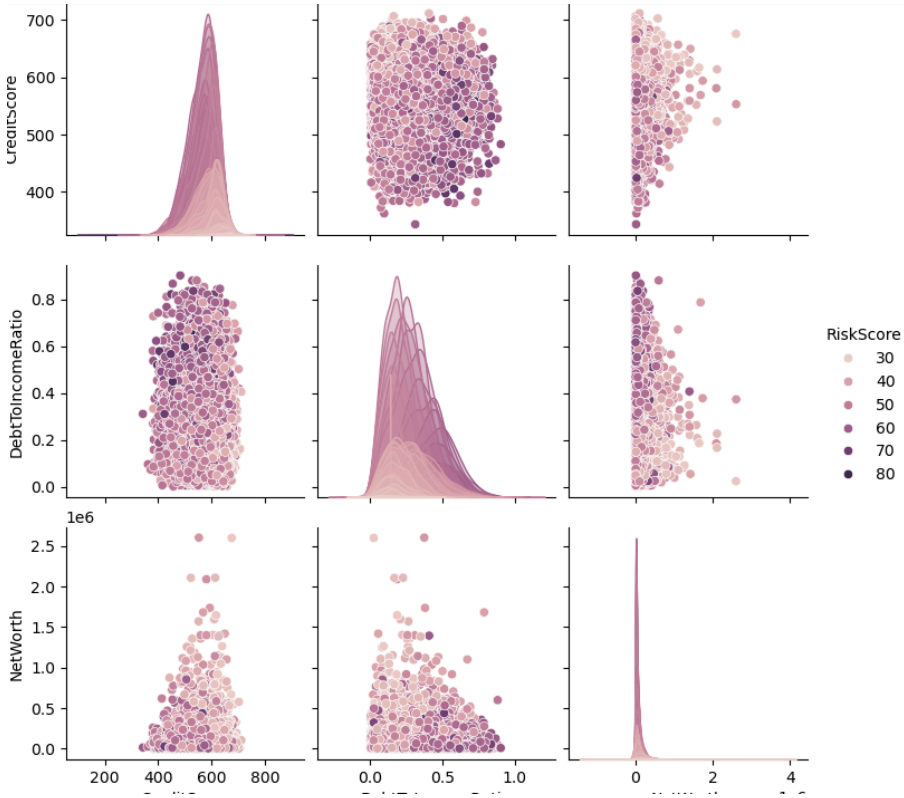
n = total number of data poinyes

y_i = actual value

$\hat{y}$ = predicted value

DATA VISUALIZATION

Fig 1



The scatter plots above show the relationship between each feature (CreditScore, DebtToIncomeRatio, and NetWorth) and how closely they are related to each other. The scatterplots do not show much, and real conclusions can be drawn. This was not surprising to me because the same thing occurred in Project 2.

PREPROCESSING

Since I used this dataset for Project 2, I followed very similar pre-processing steps. First, I dropped the ApplicationDate feature because it provided no useful information. Next, I dropped the LoanApproved feature because it was used as my target for binary classification and would provide no useful information for a Linear Regression model. Lastly, I used the *pd.get_dummies* function to change my categorical features to numerical ones.

EXPERIMENT 1

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
((16000, 44), (4000, 44), (16000,), (4000,))
```

```
model = LinearRegression()
model.fit(x_train, y_train)

print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

```
Coefficients: [-1.93796543e-02 -1.58383793e-05 -8.65810049e-03 -1.19212530e-02
  2.69182528e-05 -5.18886663e-03 -1.47036561e-02  1.22356462e-03
  4.80748127e+00 -1.37789739e-02  2.91390390e-02  1.54395535e+01
  1.31764204e+01  6.93075414e+00 -2.86899564e-02 -1.67686657e-01
 -2.12615691e-06  3.94431393e-06 -2.97129261e-06  3.28600423e-06
 -9.10217683e-04 -5.62311483e-02 -2.60325589e-02 -1.74309425e-05
  5.02703745e-09  3.13137510e+01  5.65665556e-04 -2.91712545e-01
  1.79532918e-01 -5.43526576e-03  1.04666262e-01  1.93402255e-01
  3.03828032e-01  9.76760474e-03  2.84914060e-01  1.19878600e-01
  1.98353657e-02  2.96374782e-01 -5.04762342e-01 -1.70814921e+00
  3.43203227e-01 -1.18549050e+00  2.85029262e+00  3.50906414e+00]
Intercept: 50.47614353358879
```

Evaluate


```
MSE:14.14343150812928 R-squared 0.7722014789286128 RMSE 3.760775386556512
```

I was pretty surprised with this being my first model. I predicted that it would have done a lot worse considering I have not changed anything. The R-squared value almost being at 0.80 surprised me the most and incidate that the model did well.

EXPERIMENT 2 STANDARDIZE

For my next model, I will be standardizing my data and then evaluating it. I will be standardizing my data by using the StandardScaler from sklearn.

```
scaler = StandardScaler()
```

[30]:

```
x_train_scaled = scaler.fit_transform(x_train)
```

```
x_test_scaled = scaler.transform(x_test)
x_train_scaled = scaler.fit_transform(x_train)
```

[34]:

```
model = LinearRegression()
model.fit(x_train_scaled, y_train)
```

[34]:

LinearRegression

LinearRegression()

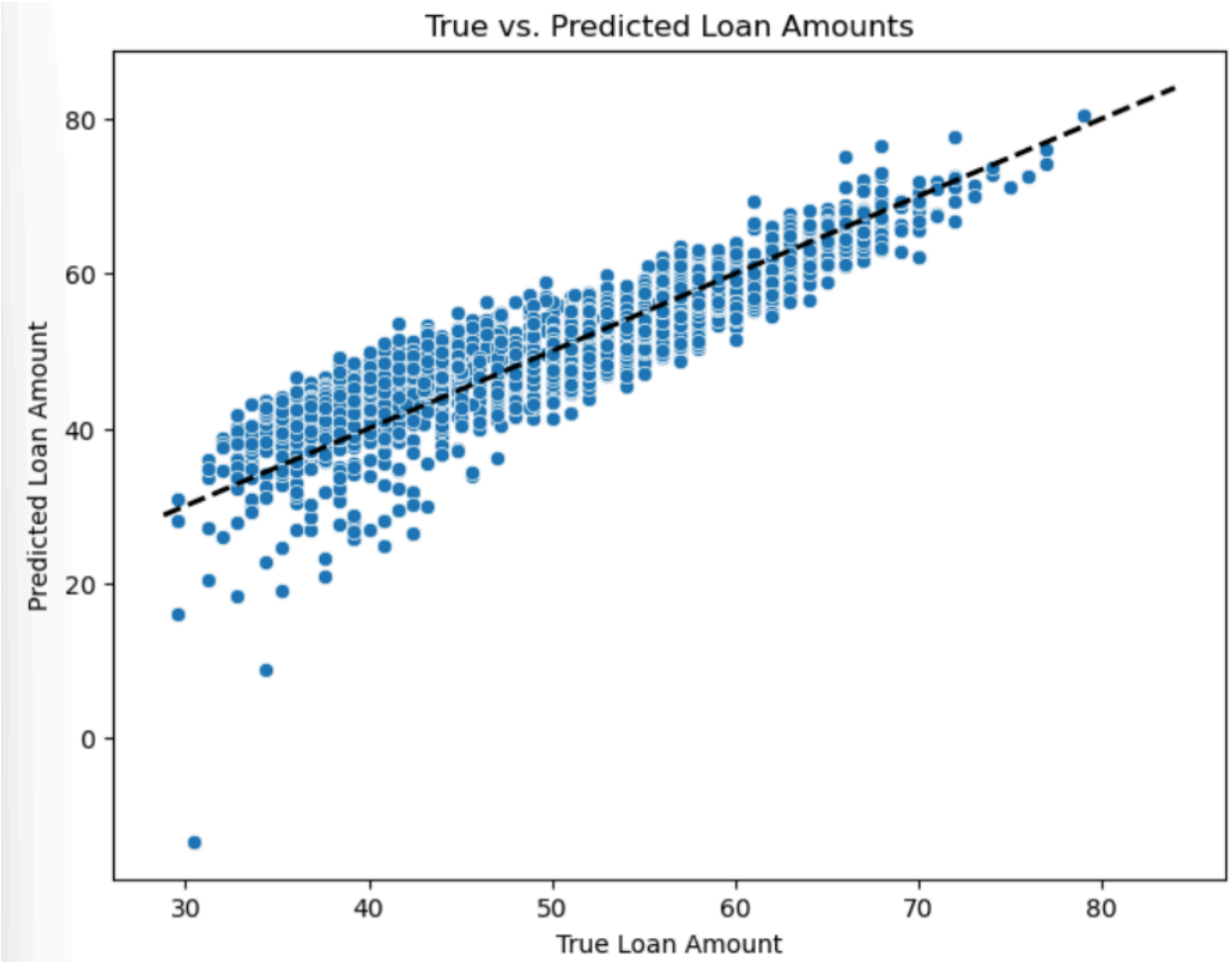
[35]:

```
y_pred = model.predict(x_test_scaled)
```

```
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

```
Coefficients: [-2.25436012e-01 -6.35081461e-01  3.48174120e+11 -1.34852162e-01
 -1.85285181e+11 -2.81815465e+11 -2.04190030e-02  2.94942986e-01
  7.69582814e-01 -2.37792355e-02  2.88650419e-02  2.47961811e+00
  2.95204870e+00  2.07572687e+00 -1.42085759e-01 -1.40404602e+00
 -1.39634436e-02  8.90330270e-03 -3.49423976e-01  1.58467276e-01
 -2.98208470e+00 -6.73987981e-03 -5.82780221e-02 -1.99739315e+00
  4.86712923e+11  1.32027968e+00  3.85991934e-01 -9.98941713e-02
  7.79654490e-02 -1.93230541e-03  4.77409040e-02  5.81150201e-02
  9.12471801e-02  3.94443866e-03  1.31280589e-01  5.99520900e-02
  9.08681536e-03  6.56914492e-02 -2.31641619e-01 -3.63975322e-01
  1.56974411e-01 -4.23840870e-01  7.69355565e-01  8.95974478e-01]
Intercept: 50.8437244888521
```

Evaluate




```
MSE:14.143353431726183 R-squared 0.7722027364515234 RMSE 3.7607650061824103
```

I was very surprised at my models' performance. I had predicted that after standardizing the data it would improve it drastically but as you can see it only improved by a very small amount. After comparing the graphs of the Linear Regression model before and after standardizing the data they almost look exactly the same.

EXPERIMENT 3 PCA

Principal Component Analysis (PCA) is a method of dimensionality prediction. Dimensionality prediction is a technique used to reduce the number of input variables while still maintaining its characteristics. PCA works by selecting features based on variance in the dataset.

[43]:

```
### Now that our data is standardized we can perform Dimensionality Reduction
pca = PCA(n_components=0.95)
x_train_pca = pca.fit_transform(x_train_scaled)
x_test_pca = pca.transform(x_test_scaled)
```

[44]:

```
model = LinearRegression()
model.fit(x_train_pca, y_train)
```

```
model = LinearRegression()
model.fit(x_train_pca, y_train)
```

[44]:

```
LinearRegression()
```

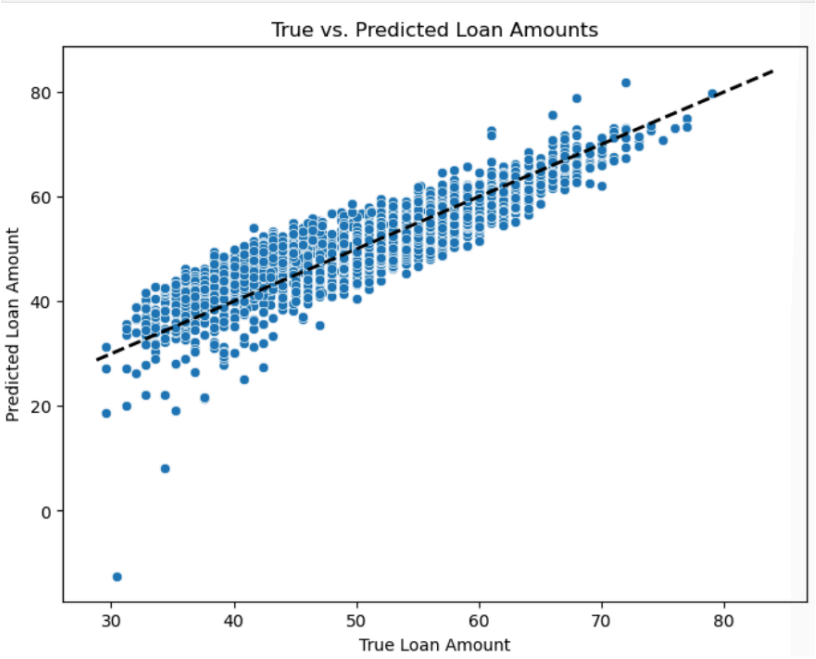
[45]:

```
y_pred = model.predict(x_test_pca)
```

```
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

```
Coefficients: [ 1.90523805  0.92949276 -0.44764088 -1.80925907 -0.96729562  0.16588617
 -0.40557482 -0.2054357   0.0207056   0.15228971  0.52560775  0.30112658
 -0.25252292  0.83835617 -0.75313014 -0.4741031  -0.2372231   0.92374769
  1.00681228 -1.14792732 -0.0332622  -1.8659319   1.86567342  0.74089321
 -1.89258647 -0.12648269 -0.03083753  0.8301761  -1.36539607 -0.22942757
  1.68545449  1.27700872  0.17733467]
Intercept: 50.84360000000001
```

Evaluate




```
MSE:14.599191682165017 R-squared 0.7648608633538914 RMSE 3.8208888602215345
```

I was very surprised after analyzing the statistics. After performing the PCA my linear regression model actually got worse by a small amount. I predicted that it would improve but it did not.

IMPACTS/CONCLUSION

The impacts of this project can allow for people to further understand their own risk score. This could allow them to be more financially stable and knowledgeable.
From this project I have learned a lot about linear regression and analying data based on statistical measures such as Mean Squared Error, R-sqaured, and Root mean squared error. Standardizing the data helped the model by a small amount. PCA caused the model's performance to decrease an insignificant amount, this could be due to an error on my part.

REFERENCES

https://seaborn.pydata.org/
https://scikit-learn.org/stable/
https://www.aporia.com/learn/root-mean-square-error-rmse-the-cornerstone-for-evaluating-regression-models/
https://www.youtube.com/watch?v=nk2CQITm_eo