



Using K-Nearest Neighbors to Classify Lending Club Loans Issued

Matt Delsman and Ian Poblete

Agenda

K Nearest Neighbors

Our Question

Lending Tree

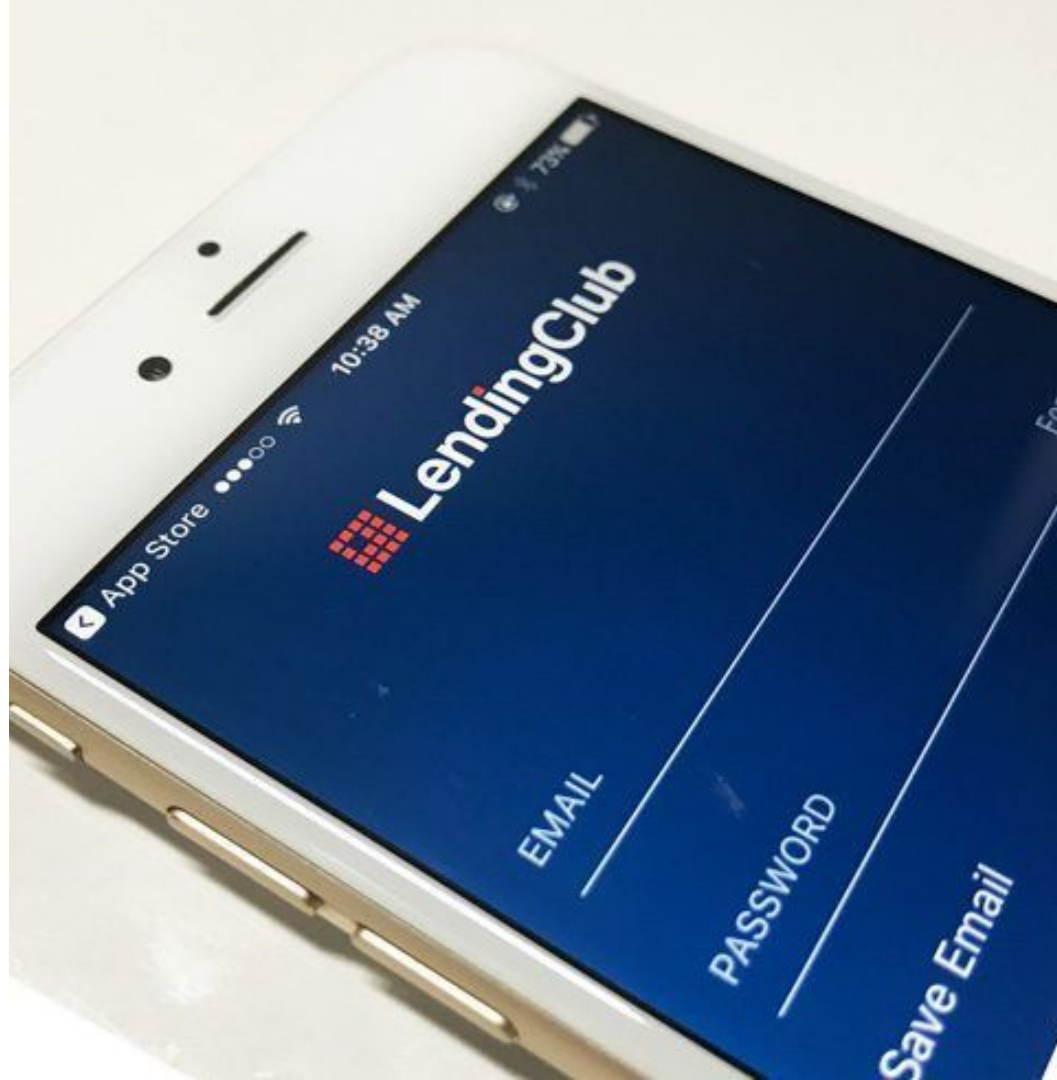
Overall Evaluation Plan

Initial Data Cleaning

Initial KNN Evaluation

Secondary Data Cleaning

Final KNN Evaluation



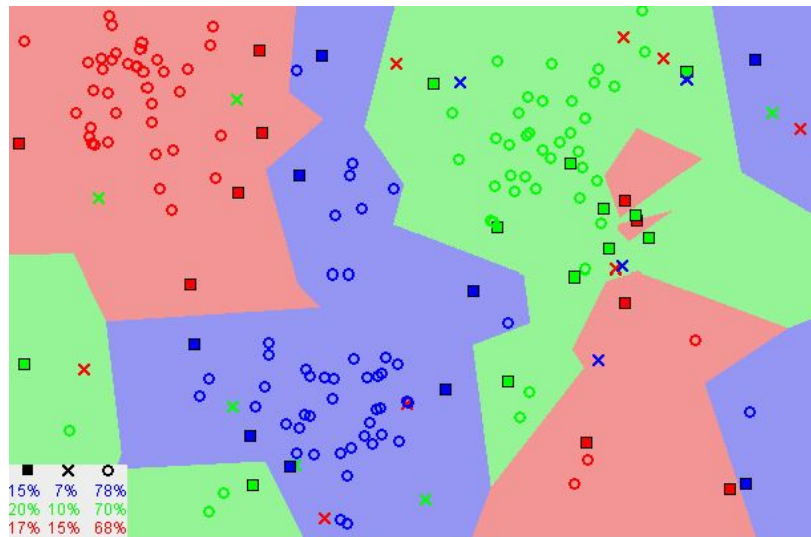
Our Formula

K-Nearest Neighbors

Grouping Data by Similar Features

K-Nearest Neighbors

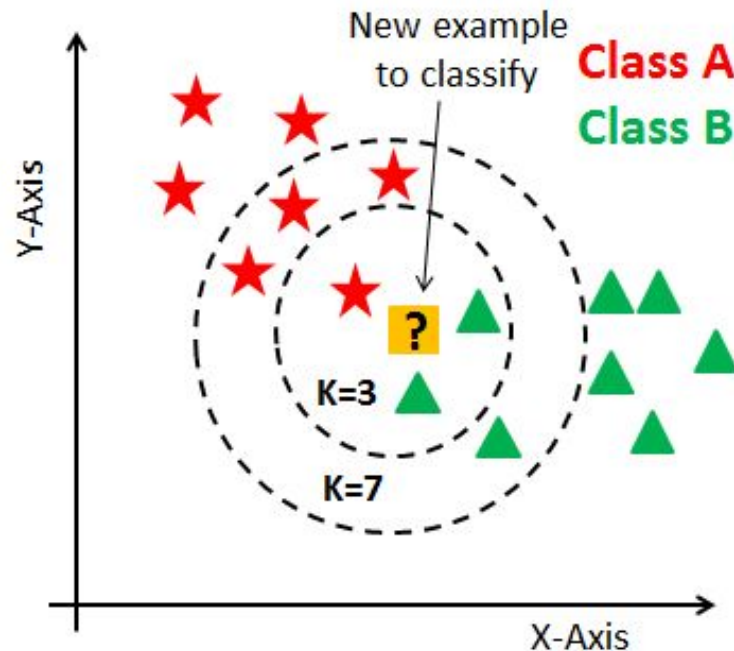
- “Similar things exist in close proximity”
- Similarity defined mathematically as distance
 - Learns classification from train set
 - Classifies a data point based on “close” other points
 - Distance types: Euclidean, Hamming, Manhattan, and Minkowski
 - Predicts based on “where” new data is



KNN Figure with 3 Classifications

K-Nearest Neighbors

- K Determination is another important step
 - K determines the number of points that are compared
 - Too small: fitting model to noise
 - Too large: misses capturing the local data structure
- Other important info
 - Non-parametric, supervised
 - Curse of dimensionality
 - Incredibly computationally intensive



KNN Prediction based on Different K's

Distance Measurements

$$D(X, Y) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Minkowski Distance Calculation

$$D_H = \sum_{i=1}^k |x_i - y_i|$$

Hamming Distance Calculation

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

Euclidean Distance Calculation

$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^n |p_i - q_i|$$

Manhattan Distance Calculation

Our Question

Lending Club Loan Classification

From 2008-2017

Backstory

- Founded in 2006
- Peer-To-Peer Lending
 - Platform where borrowers apply for loans and investors pick and choose their investments
- Loans
 - Range between \$1,000 and \$40,000
 - Standard repayment period of 3 years
- \$15.98bn lent as of December 2015
- Make their money from charging borrowers and investors a fee
 - Makes grading system incredibly important



Loan Classification

- Loans are classified into 7 Grades
 - Grades based on likelihood of repayment
- Traditional Bank - Banker with education and years of experience
- Lending Club - Amateur
 - Require help
- Grades serve as an easy reference for investors
 - Higher Grade will have a higher payback at a higher risk

Interest Rate Range		
Grade	Low	High
A	6.46%	8.81%
B	10.33%	13.08%
C	14.30%	17.74%
D	18.62%	28.80%
E	28.90%	29.00%
F	29.35%	30.75%
G	30.79%	30.99%

Loan Grades and Corresponding Interest Rates (2020)



A	B	C	D	E	F	G
Grade	Grade	Grade	Grade	Grade	Grade	Grade

7.13%

10.62%

13.91%

17.30%

20.17%

23.81%

26.24%

Average borrower interest rates as of June 30, 2016

Lower interest payments
Lower expected loan losses
(fewer charge offs)
Lower expected returns
Lower expected volatility

Higher interest payments
Higher expected loan losses
(more charge offs)
Higher expected returns
Higher expected volatility

**Can we use a KNN
model to predict
loan grade?**

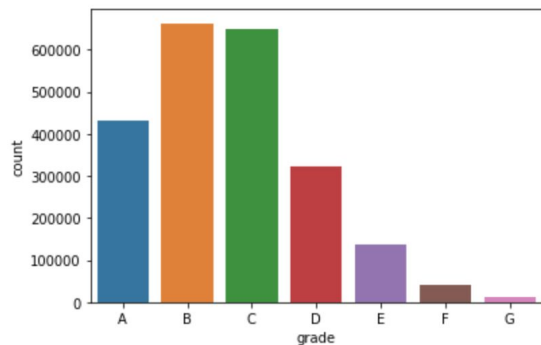
Our Data

Lending Club Loans

From 2008-2017

Our Data Set: Lending Club Loan Data

- Kaggle: <https://www.kaggle.com/wendykan/lending-club-loan-data>
- Contains data on all loans issued through Lending Club from 2007 - 2018
- 2.26 million records x 145 features (loan.csv is ~1.1 GB)
- Classify Data based on “Loan Grade” Feature
 - Loan Grade - General classification of the likelihood of a loan being paid back



Grade (Int. Rate) - Count

- A (6-8%) - 433,027
- B (10-13%) - 663,557
- C (14-17%) - 650,053
- D (18-28%) - 324,424
- E (28-29%) - 135,639
- F (29-30%) - 41,800
- G (30-31%) - 12,168

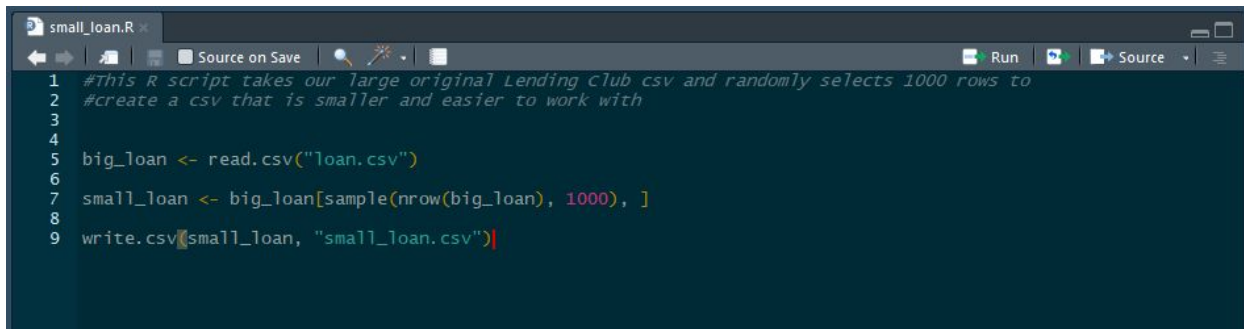
Data Refinement

- 2.26 million records x 145 features (loan.csv is ~1.1 GB)
 - Running full data set with Jupyter and sklearn takes longer than overnight
- Create smaller data sets to test code
 - Made data sets of one thousand, ten thousand, and one hundred thousand randomly sampled rows

```
In [3]: #Shape
        print(loan.shape)

        (2260668, 145)
```

Initial Data Set Shape

A screenshot of an R script editor window titled 'small_loan.R'. The window has a dark background with light-colored text. The code is as follows:

```
1 #This R script takes our large original Lending Club csv and randomly selects 1000 rows to
2 #create a csv that is smaller and easier to work with
3
4
5 big_loan <- read.csv("loan.csv")
6
7 small_loan <- big_loan[sample(nrow(big_loan), 1000), ]
8
9 write.csv(small_loan, "small_loan.csv")
```

R Code for Creating Data Subsets

Our Evaluation

Classifying Loan Grades

Grouping Data by Features

Our Evaluation Plan

Original Plan:

- Curse of Dimensionality - Need to Reduce Features
 - Dimension reduction: PCA?
 - Specific features
 - Annual income
 - Loan amount
 - Total number of credit lines
 - Loan purpose
 - DTI
 - Expected Default Rate
 - Home Status
 - Credit Inquiries in Last 12 Months
- Time Reduction - Post-Financial Crash
- Data Cleaning
- Create Model
- Evaluate

Our Evaluation Plan

Original Plan:

- Curse of Dimensionality - Need to Reduce Features
 - Dimension reduction: ~~PCA?~~
 - Specific features
 - Annual income
 - Loan amount
 - Total number of credit lines
 - Loan purpose
 - DTI
 - ~~■ Expected Default Rate~~
 - Home Status
 - ~~■ Credit Inquiries in Last 12 Months~~
- ~~● Time Reduction - Post Financial Crash~~
- Data Cleaning
- Create Model
- Evaluate

Reality:

- No Spark
- Curse of Dimensionality - Need to Reduce Features
 - Dimension reduction: **Intuition + Heat Map**
 - Specific features
 - Annual income
 - Loan amount
 - Total number of credit lines
 - Loan purpose
 - DTI
 - **Derogatory public records**
 - Home Status
 - **Credit Inquiries in Last 6 Months**
- Data Cleaning
 - **Annual income / DTI**
 - **Normalization, Encoding, Dummy Variables**
- Create Model
- **Deciding k**
- Evaluate

Variable Choice

- High number of categories in original data (145)
 - Too computationally intensive to use all
 - Reduce to most important factors
- Created a heatmap with all numeric data
 - Evaluate based on correlation
 - High correlation = same explanatory power but more noise
- Pick variables which have a low correlation
 - Did the variables we selected intuitively have low correlation?
 - Annual income, Loan amount, Total number of credit lines, Loan purpose, DTI, Expected Default Rate, Home Status, Credit Inquiries in Last 12 Months
 - ~~Expected Default Rate~~ -> **Derogatory public records**
 - ~~Credit Inquiries in Last 12 Months~~ -> **Credit Inquiries in Last 6 Months**
- Need to clean, encode, and create dummy variables



Initial Data Cleaning

We did some limited initial cleaning in order to get our code to run

- Reduce predictors, remove NA's

```
In [3]: #Clean data
#change loan to specific column
loan_s = loan[['annual_inc', 'loan_amnt', 'open_acc', 'purpose', 'dti', 'pub_rec', 'home_ownership', 'inq_last_6mths', 'grade']]

#Delete annual_inc = 0
loan_s = loan_s[loan_s.annual_inc != 0]

#Drop NAs
loan_s.dropna(inplace = True)
```

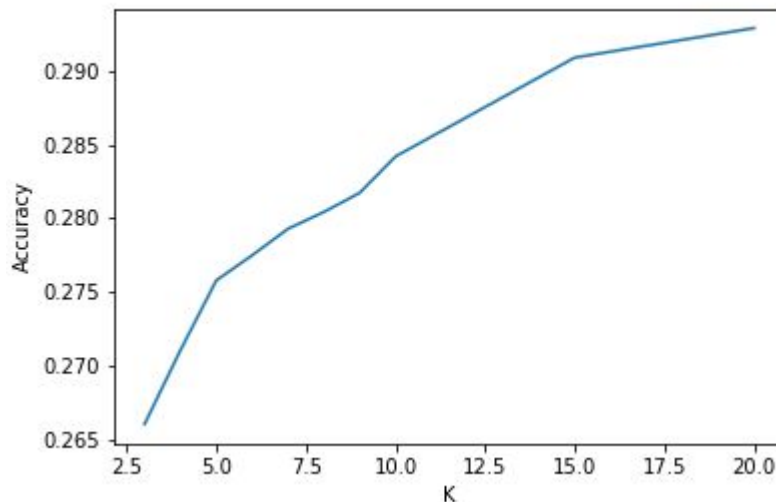
Python Code for Initial Data Cleaning

Initial Runs

Served as a baseline - removed categorical, no normalization of large data

```
Accuracy_3: 0.2660372952596991
Accuracy_4: 0.2710411315341762
Accuracy_5: 0.2757780965406812
Accuracy_6: 0.2774794008740034
Accuracy_7: 0.27928078193281514
Accuracy_8: 0.28041498482169663
Accuracy_9: 0.28171598225306066
Accuracy_10: 0.2841845414818027
Accuracy_15: 0.29088968208960203
Accuracy_20: 0.2928912165993929
```

Accuracy based on K



Accuracy Score Plotted Against K

Data Refinement

- Large Numbers
 - Annual Income - 20,000
 - DTI - 3
 - Need to normalize in order to have similar explanatory power
- Dummy variables
 - debt_consolidation
 - negatively correlated with credit_card
 - MORTGAGE negatively correlated with RENT

```
In [5]: #dummy variables for purpose, home ownership, and grade
#dummygrade = pd.get_dummies(loan['grade'])
dummpurp = pd.get_dummies(loan_s['purpose'])
dummyhome = pd.get_dummies(loan_s['home_ownership'])
#Concat
loan_s = pd.concat([loan_s, dummpurp, dummyhome], axis=1)

#drop
loan_s = loan_s.drop(['purpose', 'home_ownership'], axis=1)
```

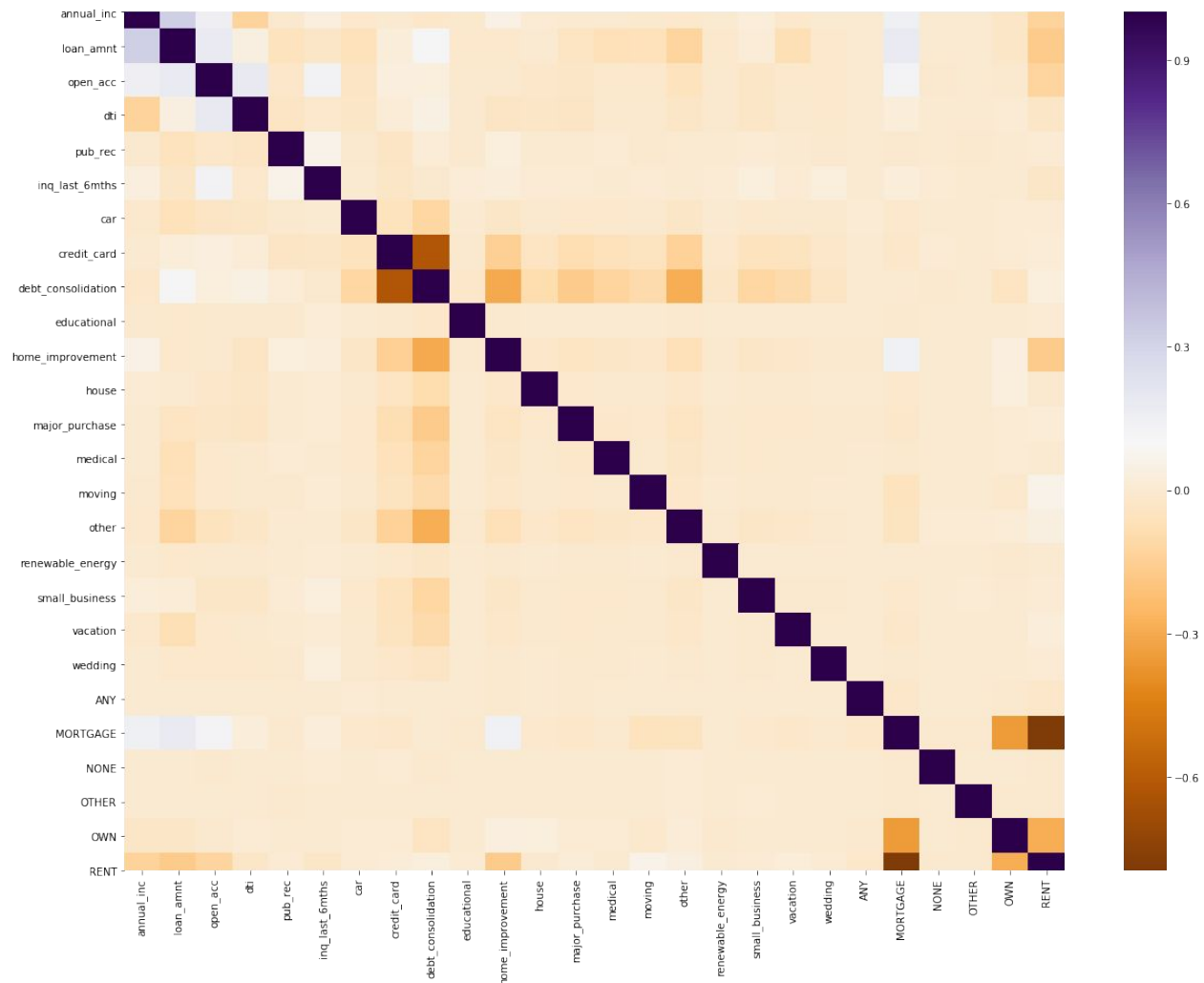
Creating Dummies for Categorical Data

```
In [7]: #Normalization

float_array = loan_s.values.astype(float)
min_max_scaler = preprocessing.MinMaxScaler()
scaled_array = min_max_scaler.fit_transform(float_array)
loan_s = pd.DataFrame(scaled_array)

print(loan_s)
```

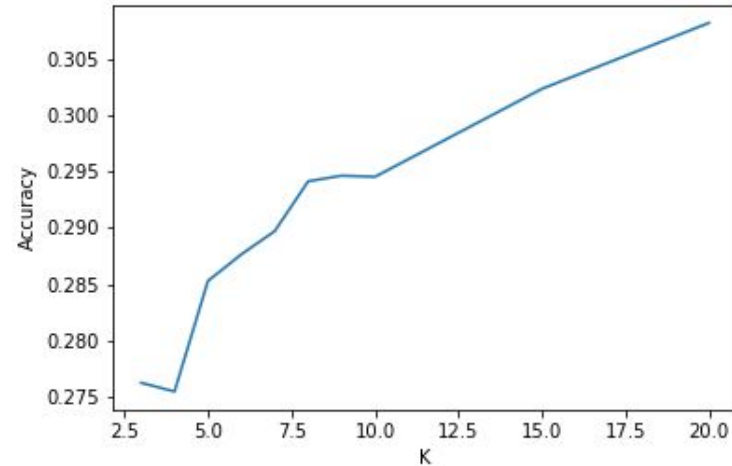
Normalization of Large Data Numbers Using Minmax Scaler



Run 2

Dummy Variables, Normalized Data = +1% Accuracy

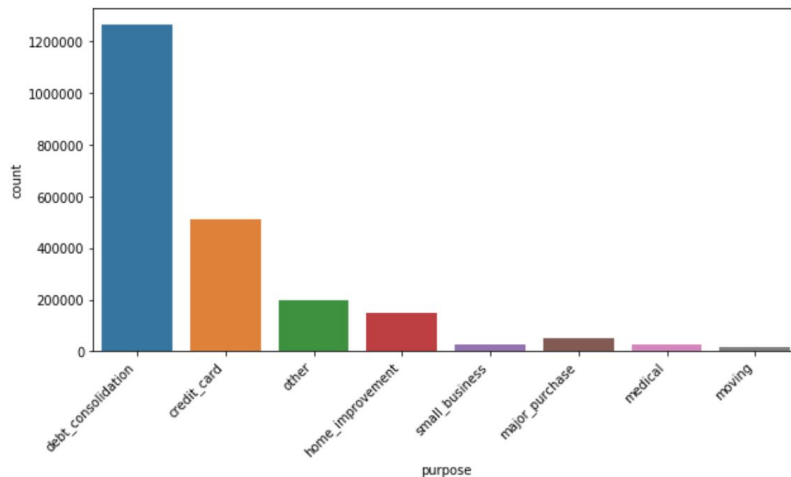
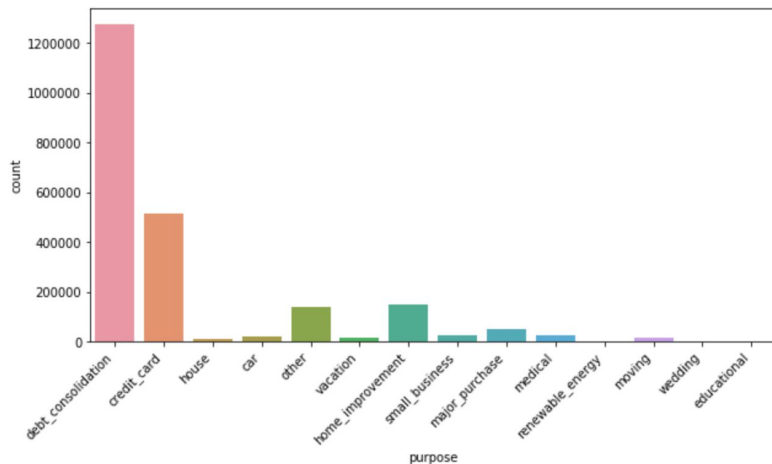
Accuracy_3: 0.27621176235113587
Accuracy_4: 0.275444507455716
Accuracy_5: 0.2852853854621877
Accuracy_6: 0.2876538679654402
Accuracy_7: 0.28972212029222405
Accuracy_8: 0.2941588551222604
Accuracy_9: 0.2946592387497081
Accuracy_10: 0.29455916202421856
Accuracy_15: 0.30239850552089936
Accuracy_20: 0.3082696734162858



Accuracy based on K

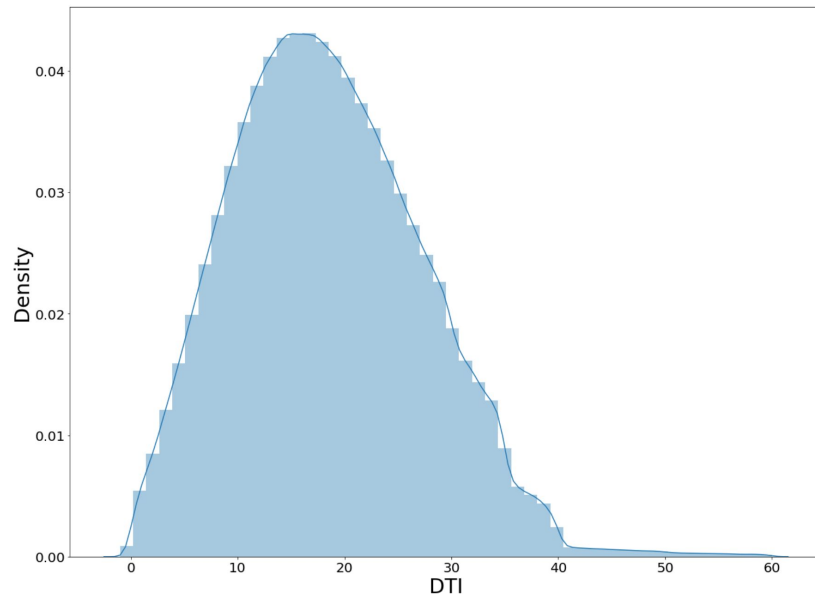
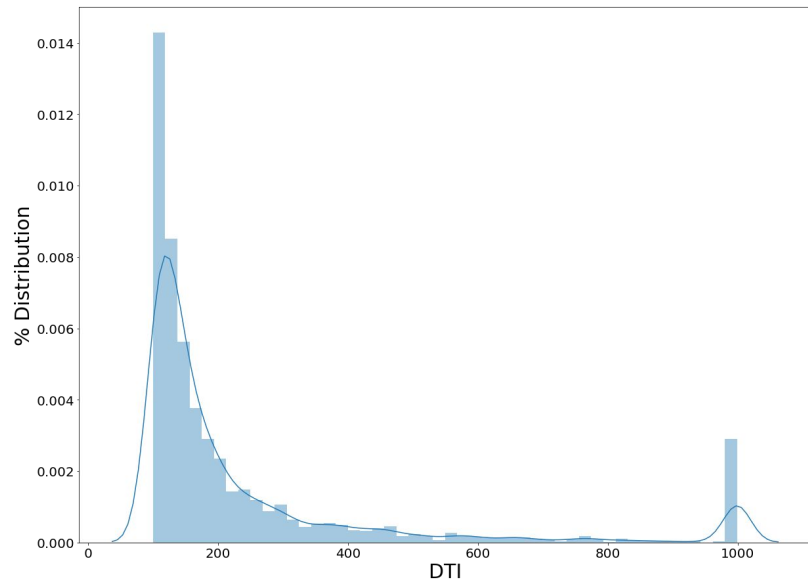
Accuracy Score Plotted Against K

Data Evaluation - Purpose



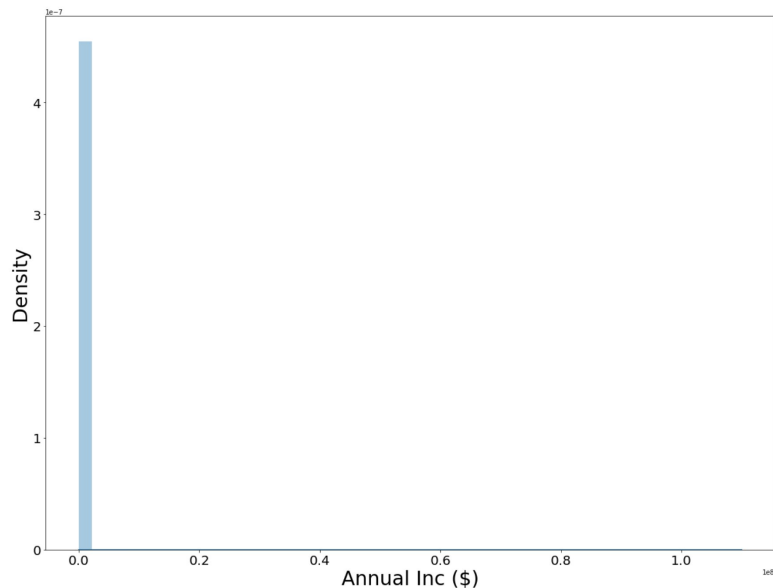
```
Purpose values house, car, vacation, renewable energy, wedding, and educational with other
loan_s.replace(to_replace = ["house", "car", "vacation", "renewable_energy", "wedding", "educational"], value = "other")
```

Data Evaluation - DTI

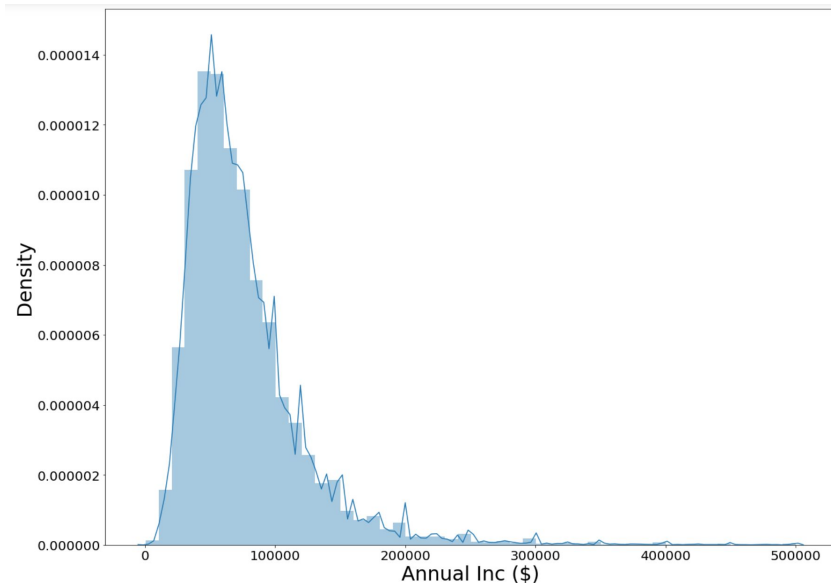


```
#Delete dti > 60  
loan_s = loan_s[loan_s.dti <= 60]
```

Data Evaluation - Annual Income



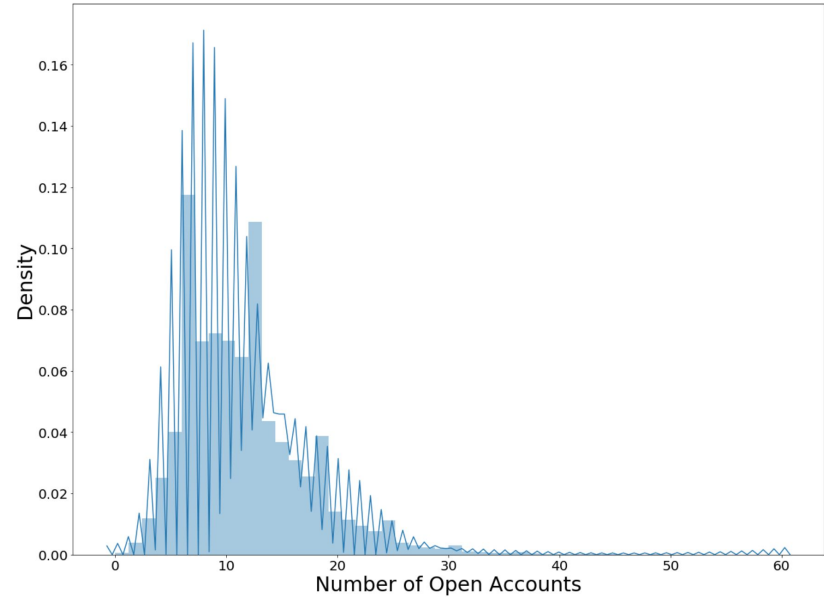
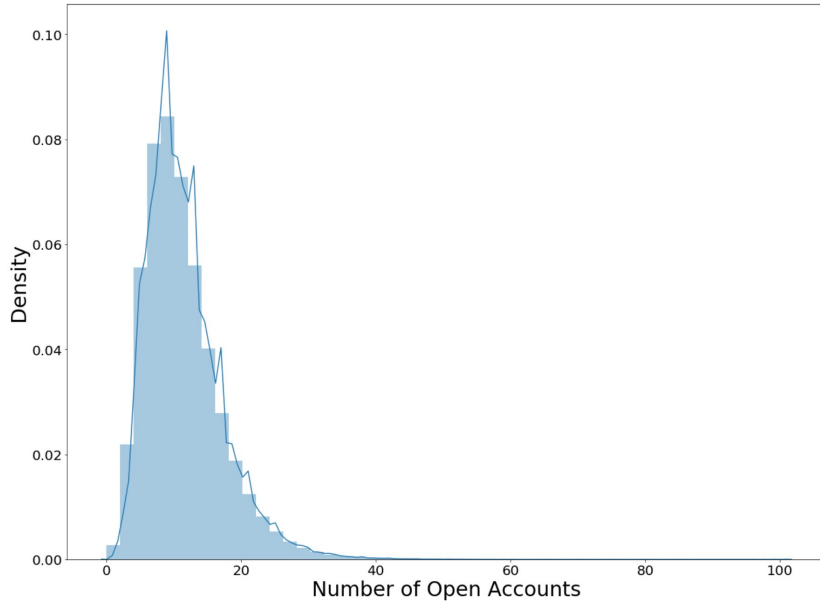
X units are in hundred millions



```
#Delete annual_inc = 0
loan_s = loan_s[loan_s.annual_inc != 0]

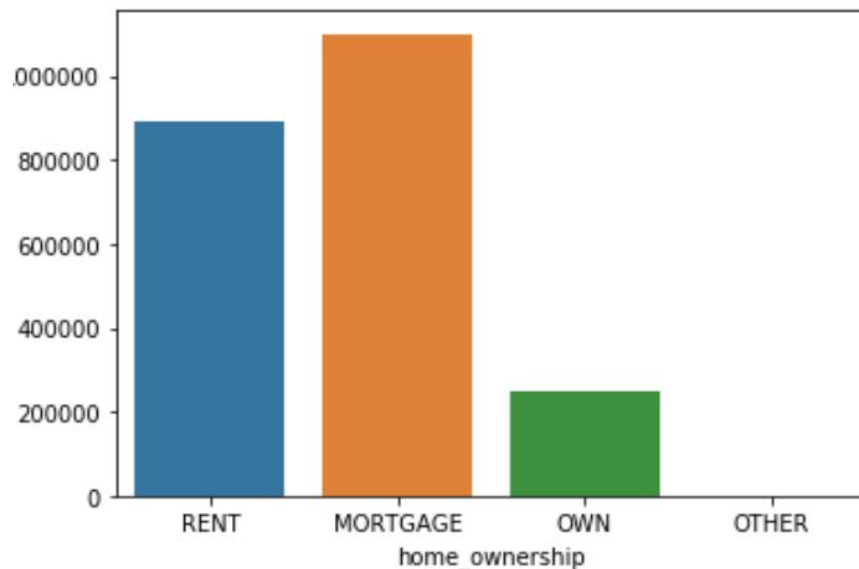
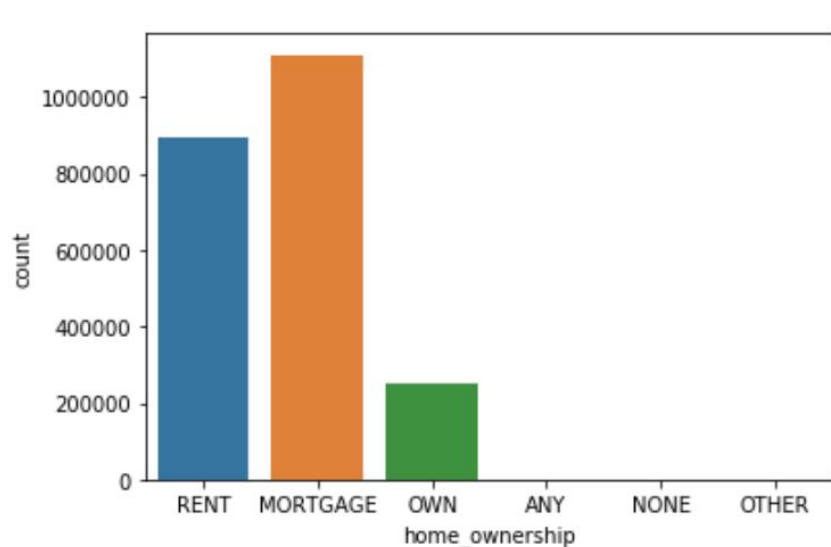
#Delete annual_inc > 500000
loan_s = loan_s[loan_s.annual_inc <= 500000]
```

Data Evaluation - Open Account



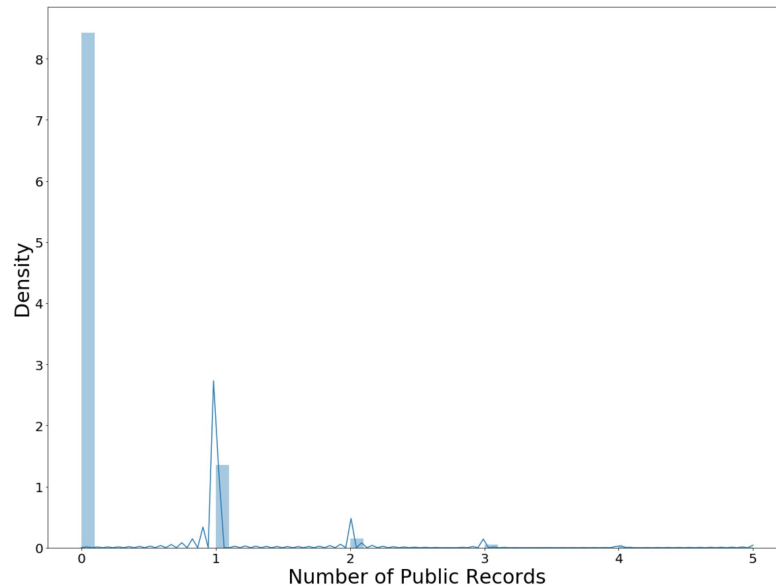
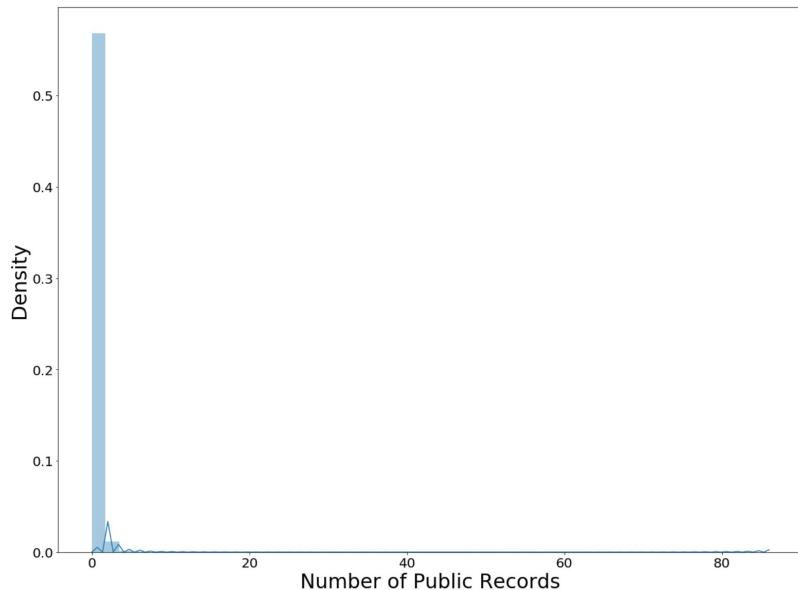
```
#Delete open acc > 60  
loan_s = loan_s[loan_s.open_acc <= 60]
```

Data Evaluation - Home Ownership



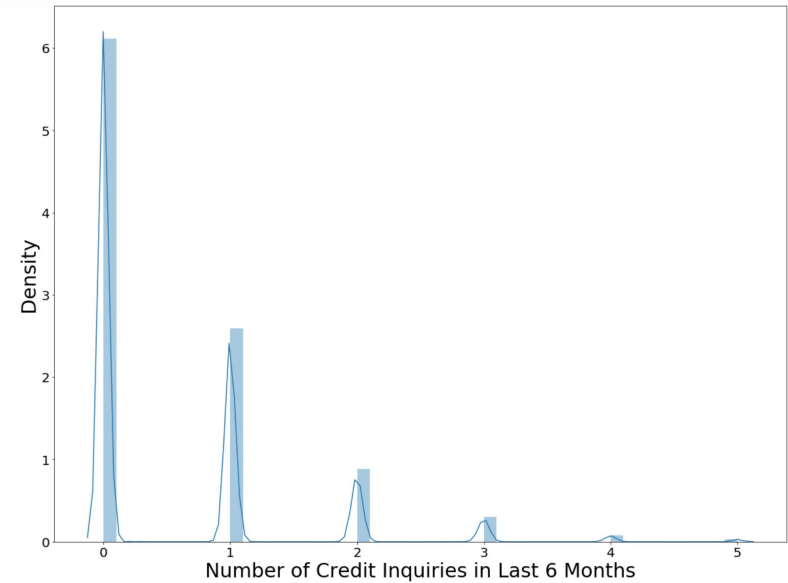
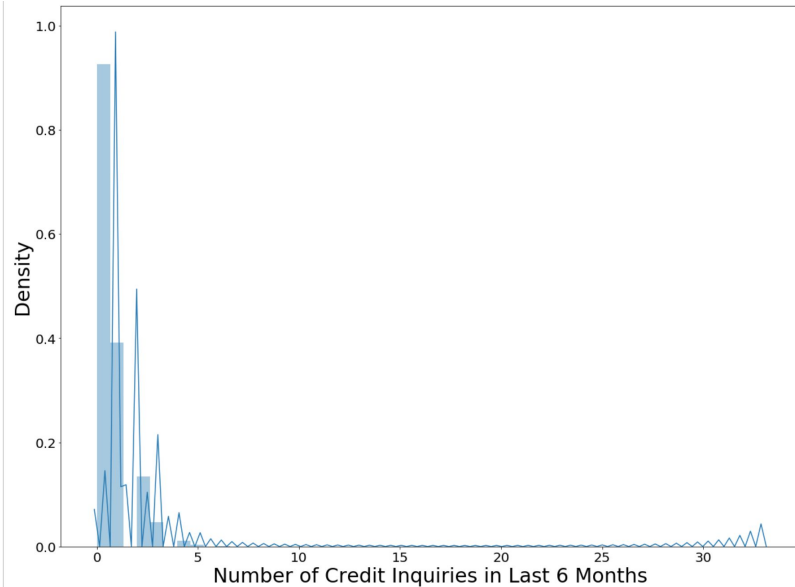
```
#Replace Home Ownership values ANY and NONE with OTHER  
loan_s = loan_s.replace(to_replace = ["ANY", "NONE"], value = "OTHER")
```

Data Evaluation - Public Records



```
#Delete pub rec > 5  
loan_s = loan_s[loan_s.pub_rec <= 5]
```

Data Evaluation - Inq last 6 months



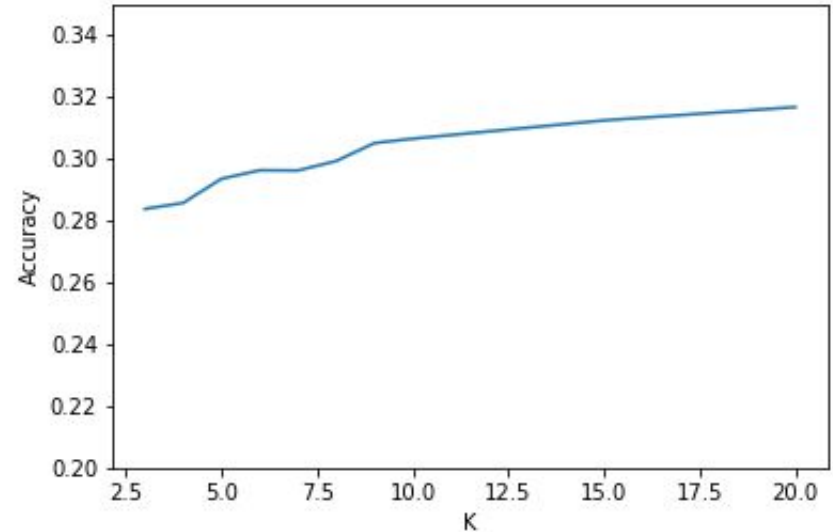
```
#Delete inq_last_6mths > 5  
loan_s = loan_s[loan_s.inq_last_6mths <= 5]
```

Run 3

Removed all Outliers = +1% Accuracy, smaller spread

```
Accuracy_3: 0.2837238971328812
Accuracy_4: 0.2856711206607131
Accuracy_5: 0.29346001477204053
Accuracy_6: 0.2962129859665615
Accuracy_7: 0.2961458403276707
Accuracy_8: 0.29920096689720005
Accuracy_9: 0.3050090646612503
Accuracy_10: 0.3064191230779561
Accuracy_15: 0.3123279393003424
Accuracy_20: 0.3166252601893507
```

Accuracy based on K

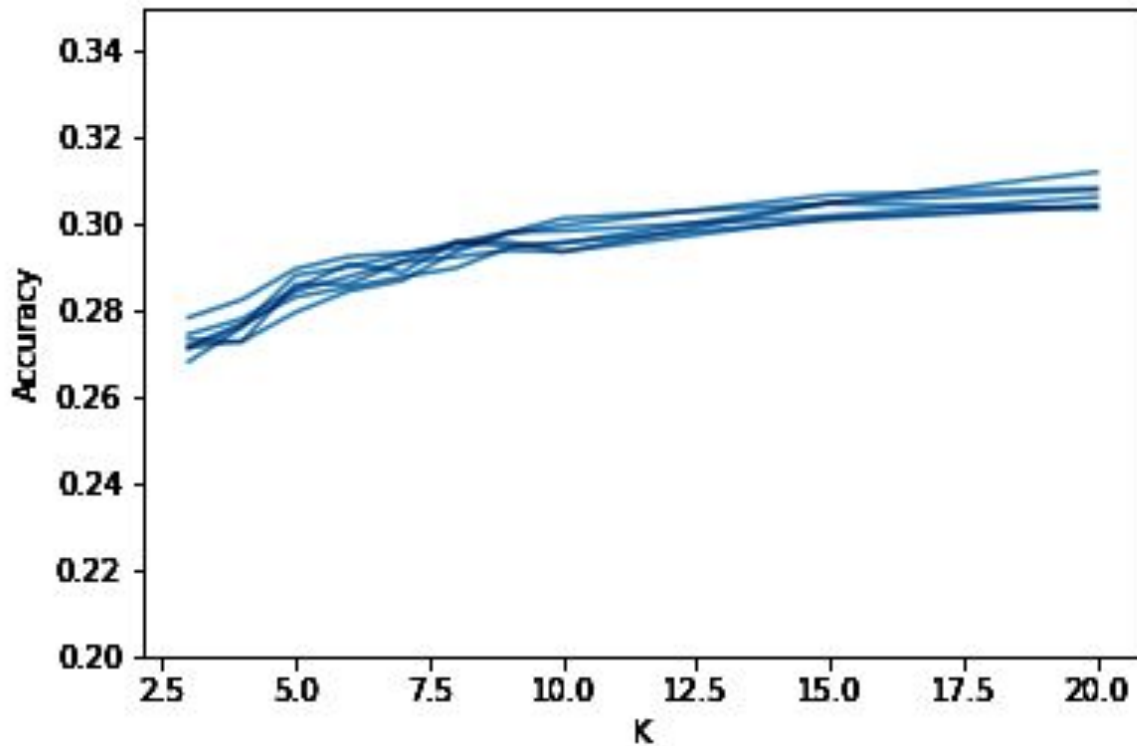


Accuracy Score Plotted Against K

Deciding k

After $K = 11$, there were diminishing returns on accuracy

Higher K values lead to significant processing times, not worth it



Last Gasp

Trying Different Distances

```
In [10]: #Specify number of neighbors
```

```
#Default with Minokowski p = 2 (Euclidean)  
knn_11_p2 = KNeighborsClassifier(n_neighbors = 11)
```

```
#Default with Minokowski p = 1 (Manhattan)  
knn_11_p1 = KNeighborsClassifier(n_neighbors = 11, p = 1)
```

```
In [11]: #Train model with test set
```

```
knn_11_p2.fit(x_train, y_train)  
knn_11_p1.fit(x_train, y_train)
```

```
Out[11]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',  
                             metric_params=None, n_jobs=None, n_neighbors=11, p=1,  
                             weights='uniform')
```

```
In [12]: #Predict output with test set
```

```
y_pred_11_p2 = knn_11_p2.predict(x_test)  
y_pred_11_p1 = knn_11_p1.predict(x_test)
```

```
In [13]: #Model Accuracy
```

```
from sklearn import metrics  
Accuracy_11_p2 = metrics.accuracy_score(y_test, y_pred_11_p2)  
Accuracy_11_p1 = metrics.accuracy_score(y_test, y_pred_11_p1)  
print("Accuracy_11_p2:", Accuracy_11_p2)  
print("Accuracy_11_p1:", Accuracy_11_p1)
```

```
Accuracy_11_p2: 0.3104257458933959
```

```
Accuracy_11_p1: 0.3043915521287295
```

Finalized Result

```
In [13]: #Model Accuracy
          from sklearn import metrics
          Accuracy_11 = metrics.accuracy_score(y_test, y_pred_11)

          print("Accuracy_11:", Accuracy_11)
```

```
Accuracy_11: 0.31210512864708084
```

Possible Explanations:

Too many predictors, too many categories, human grade classification

Finalized Result - Individual Grade Comparisons

```
#Model Accuracy
from sklearn import metrics
Accuracy_11 = metrics.accuracy_score(y_test, y_pred_11)

print("Accuracy_A_vs_B:", Accuracy_11)
```

Accuracy_A_vs_B: 0.5986977182530576

```
#Model Accuracy
from sklearn import metrics
Accuracy_11 = metrics.accuracy_score(y_test, y_pred_11)

print("Accuracy_A_vs_C:", Accuracy_11)
```

Accuracy_11: 0.661488382186536

```
#Model Accuracy
from sklearn import metrics
Accuracy_11 = metrics.accuracy_score(y_test, y_pred_11)

print("Accuracy_A_vs_D:", Accuracy_11)
```

Accuracy_A_vs_D: 0.7186920777942337

```
#Model Accuracy
from sklearn import metrics
Accuracy_11 = metrics.accuracy_score(y_test, y_pred_11)

print("Accuracy_A_vs_E:", Accuracy_11)
```

Accuracy_A_vs_E: 0.8169363264486957

```
#Model Accuracy
from sklearn import metrics
Accuracy_11 = metrics.accuracy_score(y_test, y_pred_11)

print("Accuracy_A_vs_F&G:", Accuracy_11)
```

Accuracy_A_vs_F&G: 0.9083001337174839

Grade Comparison	Accuracy
A vs B	59.86%
A vs C	66.15%
A vs D	71.87%
A vs E	81.69%
A vs F&G	90.83%

Appendix

Feature	Column Name	Data Type
Annual Income	annual_inc	Float
Loan Amount	loan_amnt	Integer
Open Accounts	open_acc	Integer
Loan Purpose	purpose	Character
DTI (Monthly Debt Payment/Total Debt)	dti	Float
Derogatory Public Records	pub_rec	Integer
Home Ownership	home_ownership	Character
Credit Inquiries in last 12 Months	inq_last_6mths	Integer
Grade	grade	Character

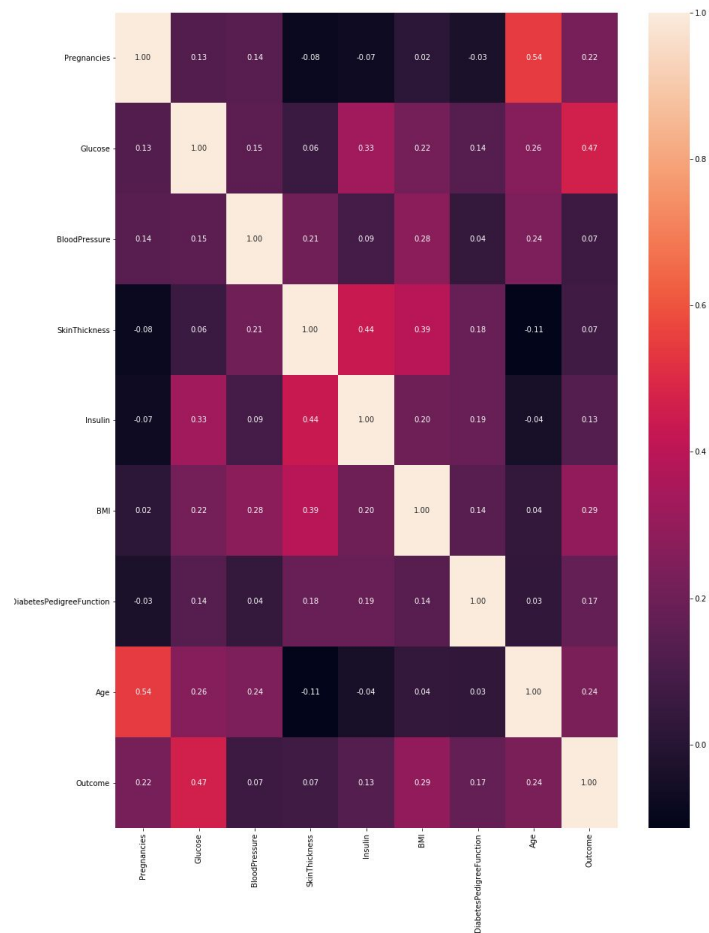
References

<https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761>
https://en.wikipedia.org/wiki/Euclidean_distance
https://en.wikipedia.org/wiki/Taxicab_geometry
https://en.wikipedia.org/wiki/Kernel_density_estimation
<https://stackoverflow.com/questions/19960077/how-to-filter-pandas-dataframe-using-in-and-not-in-like-in-sql>
https://www.saedsayad.com/k_nearest_neighbors.htm
https://en.wikipedia.org/wiki/Minkowski_distance
<https://www.youtube.com/watch?v=Eecjn0Uirw>
<https://en.wikipedia.org/wiki/LendingClub>
<https://www.lendingclub.com/foiofn/rateDetail.action>
<https://www.drawingfromdata.com/how-to-rotate-axis-labels-in-seaborn-and-matplotlib>
<https://discuss.codecademy.com/t/in-seaborn-can-we-set-a-specific-bar-order-for-a-barplot/354499>
<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html>
<https://www.moneycrashers.com/lending-club-review-peer-to-peer-lending/>
<https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/>
<https://towardsdatascience.com/better-heatmaps-and-correlation-matrix-plots-in-python-41445d0f2bec>

OLD SLIDES

Our Evaluation Plan

- Proposed Evaluation Visualization
 - Heat Map
- Libraries we plan on using
 - Sklearn.neighbors
 - Sklearn.metrics
 - Pylab
 - Matplotlib
 - seaborn
- Model Evaluation
 - Using Accuracy from Sklearn.neighbors package
- Overall Goal
 - Most important Features for each Loan Grade



Agenda

Data Overview

Question you are looking to answer

Data Refocus - Original Focus + No Spark, create smaller set in order to make program run

Variable Choice - Heat Map + Intuition

Data Cleaning - Data Cleaning N/A's + 0's Annual Income

Data Cleaning - Dummy Variables + Encoding + Final heat map

Accuracy - Normalization

Deciding K - Overlap all the Graphs

Visualized Results - Accuracy Plot + Explanation + Reason for low + suggestions (SVM)

Final Result - Pick a K and Run for Big Data Set

Distance Measurements

- Hamming

$$D_H = \sum_{i=1}^k |x_i - y_i|$$

- Minkowski

$$D(X, Y) = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{\frac{1}{p}}$$

https://www.saedsayad.com/k_nearest_neighbors.htm

https://en.wikipedia.org/wiki/Minkowski_distance

https://www.youtube.com/watch?v=_EEcn0Uirw

Variable Choice

- Created a heatmap with all numeric data
- Did the variables we selected intuitively have low correlation?
 - Annual income
 - Loan amount
 - Total number of credit lines
 - Loan purpose
 - DTI
 - ~~Expected Default Rate~~ -> **Derogatory public records**
 - Home Status
 - ~~Credit Inquiries in Last 12 Months~~ -> **Credit Inquiries in Last 6 Months**
- Need to clean, encode, and create dummy variables



Data Exploration

- Data shape
 - Df.shape

```
In [12]: print(loan.shape)
          print(loan_s.shape)

(2260668, 145)
(2258919, 26)
```

- More data visualization
- Test with different distance measurement