Ivan Escalona-Faria
Nikhil Gupta
Anusha Palisetty

Brendan Blake Camp
CSC 4740
10 December 2019

Final Project Report

Machine Learning and Data Mining - along with Artificial Intelligence - were once the promising future of Computer Science. It is safe to say that today it looks more than just promising, and a little bit closer to the present than to the future. Thanks to the advancement in storing technologies, and computational power, what was once theoretical foundations for extracting relevant insights from Data; and being able to predict new information based on historical data, is what's driving today's business decisions, what is the hope for medical advancements, understanding complex data such as that from space, image recognition, and many other interesting and relevant applications use Data Mining/Machine Learning algorithms as their core and driving force.

In this blog post we are going to try to go through all the steps that involve a machine learning/data mining workflow right from the start, when we make the decision of which data set to use, all the way to the conclusions and further recommendations after we have evaluated the data.

First let's begin by trying to define the concepts. Data Mining and Machine Learning are terms which some might even use interchangeably to describe processes and activities that are . One way of looking at it is that both are meant to obtain insights from data, the difference is that the Data Mining will be done by the user actively applying functions and algorithms to obtain insights, while Machine Learning the user allows the model to do the predictions or analysis by itself.

In any case, the workflow can be outlined in different ways, but the high level idea is mostly about the following:

1) Business understanding
2) Data understanding
3) Data preparation
4) Modeling
5) Evaluation
6) Deployment.

(Source) - CRISP-DM: Cross Industry Standard Process Data Mining

Out of these one would think that the most important and the ones that one should spend more time on are the modeling, or evaluation parts. But as it turns out, the more lengthy - and sometimes tedious - part is Data Preparation, but even this can't be done unless there is accurate business understanding and the subsequent data understanding.

There is an expression "**Garbage In, Garbage Out**". This means that if the quality of the input is poor, then the results and insights derived out of it are more probably equally poor. Is for this that the most important steps are the Business and Data understanding, followed by the data preparation.

The first thing we needed to do was to decide which data set to use, this can be tricky because of one of the most important things that are needed to be able to derive relevant insights from data, that is "**Domain knowledge**". It is the understanding, skills, insightfulness, and information on a particular subject, which is crucial to asking the right questions, and interpreting the data at hand.

The first thing was to decide on a data set to work with. We chose the earthquake data set because it would be interesting trying to predict earthquakes based on historical data, perhaps regions of high occurrence, time periods of high incidence, etc. However, when taking a closer look at the data, it is seen that most of the features are of the nominal type, mostly ID's and dates, which can't be easily used in descriptive or predictive models. This is telling because it shows that even if it is a good idea, the data needs to have enough quality, and quantity to be of use and applied to models.

After this, we pursued another data set, which was richer in features, and instances. We used a loan data set. This data set was pulled from the kaggle.com website:

https://www.kaggle.com/wendykan/lending-club-loan-data

Lending club is the largest peer-to-peer marketplace connecting borrowers with lenders. Borrowers apply through an online platform where they are assigned an internal score. Lenders decide 1) whether to lend and 2) the terms of loan, such as interest rate, monthly instalment, tenure etc. Some popular products are credit card loans, debt consolidation loans, house loans, car loans etc.

There are many features that contain a widely varied and can be classified into three categories:

1) Customer's demographics: such as employment length, title, annual income, zip code, etc.

2) Loan information: such as loan amount, funded amount, interest rate, loan status, loan grade, etc.
3) Characteristics, and customer behavior variables: Credit history information, loan purpose, application type.

One crucial step to understand the data is to perform univariate and bivariate analysis by means of visualizations. The pairplot and the correlation matrix shows that there are very few variables that have a linear relation, only the installment vs funded amount, which of course makes sense since the installment will be proportional to the amount that was borrowed. Then the interest rate with the given grade, which again makes sense because there will be a strong correlation as lenders typically will give lower interest rates to those who are better qualified.

So, with this overview what can we predict?

From the point of view of the customer, the grade they might receive will affect their interest rate on the loan, so let’s try to predict the grade of the customers based on their data and see if that can be done. From the data on the grades we can see that there are seven different grades that can be given to a customer - A through F with A being the highest - this presents a problem because models tend to give better results the less number of targets are in the mix, what’s called “cardinality”. So these were encoded into three separate categories (High, Medium, Low) to help improve this. It should be pointed out that there are some subgrades as well, but this can be worked in a similar manner if needed.

The next step is to continue to clean and prepare the data. Three major things were done: Removing the null values; binning numerical into categorical data, and of course one hot encoding which was done for the grades.

Removing Null Values:

The null values are points with missing records for a given feature on a given row. These cannot be added as input to the model and this becomes a problem which has to be carefully handled. Ideally by removing the null values, we will still have enough information to get a representative sample of the original data.

First we dropped the columns which had only null values, which also reduces the dimensionality of the data from 147 features to 65 features, which as we will explain later actually helps decrease the computational cost of running the more computationally exhaustive models such as KNN.

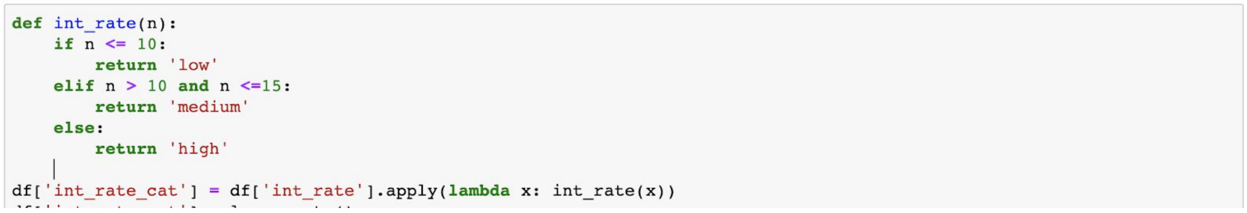
```python
#Dropping columns having all NA values
blankColumns = df.columns[(df.isnull().sum() == df.shape[0])]
df = df.drop(blankColumns, axis=1)
```

This still leaves columns with a great amount of null values, so columns with more than 60% of null values were removed. This threshold is somewhat arbitrary, but provides a reasonable cutting point which can help reduce the number of null values, this takes the number of features down to 56.

After this there are still some columns with null values, one option could be to impute values such as fill with zeros, or even the mean, median or other central tendency values. Another could be to remove only those rows that have null values, but this might dramatically reduce the number of records present in the data. The alternative that we used was to analyze the features and decide which ones should be the ones to be used, leaving the ones that are typically more directly affecting the possible grade that a borrower obtains. This serves the purpose of making the model less computationally demanding, as some of the less greedy algorithms will take more time to compute as the number of features increase. This is sometimes called “The curse of dimensionality”. A more realistic approach in which computational power is not a constraint can certainly include all the features, as long as they are properly preprocessed, and this might provide better results overall.

Converting Numerical to Categorical:

Once the features were selected, those with categorical data were transformed into numerical. Again this is because the models are not typically built to receive “strings” as input, but rather numerical data.

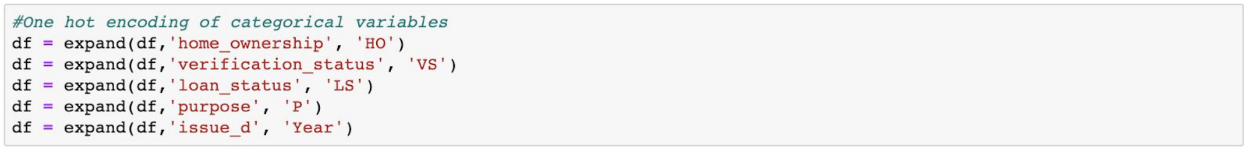


```
def int_rate(n):
    if n <= 10:
        return 'low'
    elif n > 10 and n <=15:
        return 'medium'
    else:
        return 'high'

df['int_rate_cat'] = df['int_rate'].apply(lambda x: int_rate(x))
```

One hot encoding:

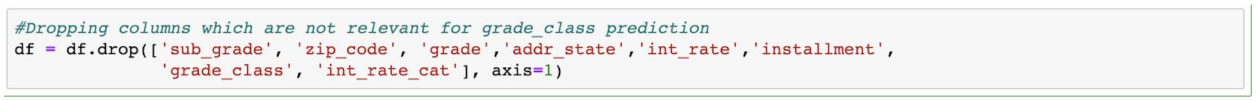
When presented with categorical data that is not binary, all sorts of problems occur when running the model, one way of addressing this is to map them as binary by means of a process called “One-hot encoding”, which consists in visualizing the different labels as either you are label x or not, instead of labelling it as t,u,v,x,y,z. This will allow for the models to be more robust.



```
#One hot encoding of categorical variables
df = expand(df,'home_ownership', 'HO')
df = expand(df,'verification_status', 'VS')
df = expand(df,'loan_status', 'LS')
df = expand(df,'purpose', 'P')
df = expand(df,'issue_d', 'Year')
```

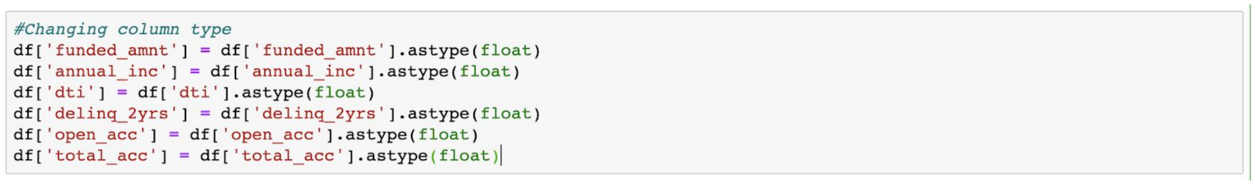
Classification Model:

After the data cleaning is completed we prepare the dataset for modelling. The unwanted columns which are not relevant are dropped and columns which are useful for prediction are included in the dataset.



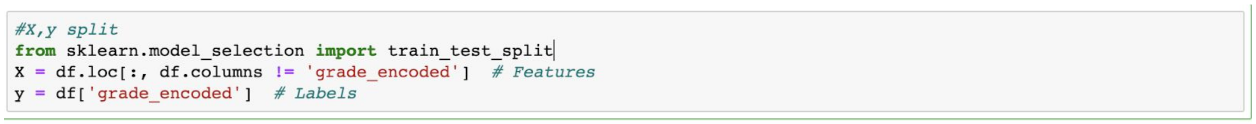
```python
#Dropping columns which are not relevant for grade_class prediction
df = df.drop(['sub_grade', 'zip_code', 'grade','addr_state','int_rate','installment',
              'grade_class', 'int_rate_cat'], axis=1)
```

All the numerical columns are type casted into float data type.

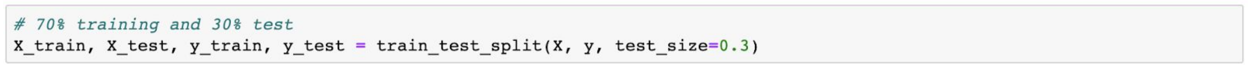


```python
#Changing column type
df['funded_amnt'] = df['funded_amnt'].astype(float)
df['annual_inc'] = df['annual_inc'].astype(float)
df['dti'] = df['dti'].astype(float)
df['delinq_2yrs'] = df['delinq_2yrs'].astype(float)
df['open_acc'] = df['open_acc'].astype(float)
df['total_acc'] = df['total_acc'].astype(float)
```

Now we create two arrays X and y where X contains the descriptive features used for prediction and y contains the class labels of the samples. To partition the data into training and test subsets ‘train_test_split’ function is used from scikit learn’s model_selection model. By setting test_size=0.3 we assign 30% of samples to X_test and y_test while the remaining 70% of the data is assigned for X_train and y_train.



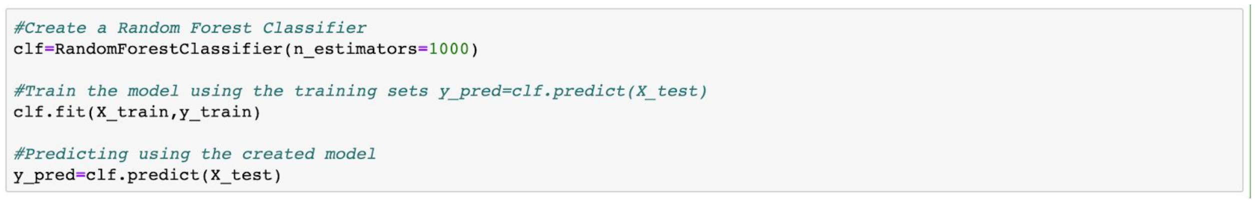
```python
#X,y split
from sklearn.model_selection import train_test_split
X = df.loc[:, df.columns != 'grade_encoded']  # Features
y = df['grade_encoded']  # Labels
```



```python
# 70% training and 30% test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

The training data is fed to classification models like Random Forest, Gaussian Naive Bayes, SVM, Logistic Regression, KNN, AdaBoost and prediction for the Grade is performed.

Random Forest Classifier gave an accuracy of 63%



```python
#Create a Random Forest Classifier
clf=RandomForestClassifier(n_estimators=1000)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)

#Predicting using the created model
y_pred=clf.predict(X_test)
```

Gaussian Naive Bayes Classifier gave an accuracy of 58%

```
#Create a Gaussian Classifier
gnb = GaussianNB()

#Train the model using the training sets
gnb.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = gnb.predict(X_test)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Support Vector Machine gave an accuracy of 53%

```
#Create a svm Classifier
clf = svm.SVC(gamma='scale') # Linear Kernel

#Train the model using the training sets
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)

# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Logistic Regression gave an accuracy of 55%

```
#Logistic Regression
model = LogisticRegression(solver = 'lbfgs')
model.fit(X_train, y_train)
# use the model to make predictions with the test data
y_pred = model.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

K-Nearest Neighbours has an accuracy of 72 % and AdaBoost Classifier gave an accuracy of 68%.

```
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X, y)
y_pred = neigh.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7216912081288211
```

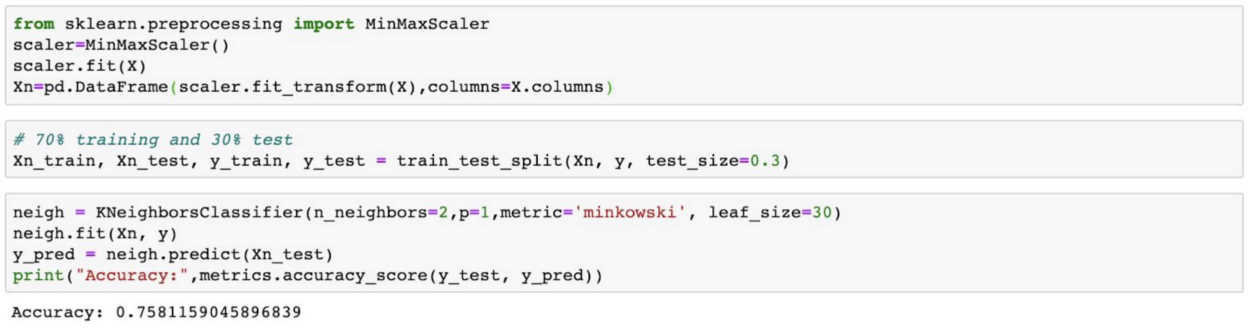
```
clf = AdaBoostClassifier(n_estimators=1000, random_state=0)
clf.fit(X, y)
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
        learning_rate=1.0, n_estimators=1000, random_state=0)
clf.feature_importances_

y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

From the models we observe that KNN classifier provided the highest accuracy with 72%. As seen, there wasn't a systematic or methodical way to develop the models, this of course would have provided better chances of tuning to the best parameters for each model in order to improve it in place. This can be a recommendation that comes out of this project.

Better Performance:
In order to provide better performance we normalize the data first and then feed the data to the classification models. Range normalization is performed on the descriptive features using

MinMaxScaler from preprocessing model of Sklearn. After scaling we observe the accuracy is increased to 76% using KNN classification.



```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
scaler.fit(X)
Xn=pd.DataFrame(scaler.fit_transform(X),columns=X.columns)
```

```
# 70% training and 30% test
Xn_train, Xn_test, y_train, y_test = train_test_split(Xn, y, test_size=0.3)
```

```
neigh = KNeighborsClassifier(n_neighbors=2,p=1,metric='minkowski', leaf_size=30)
neigh.fit(Xn, y)
y_pred = neigh.predict(Xn_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

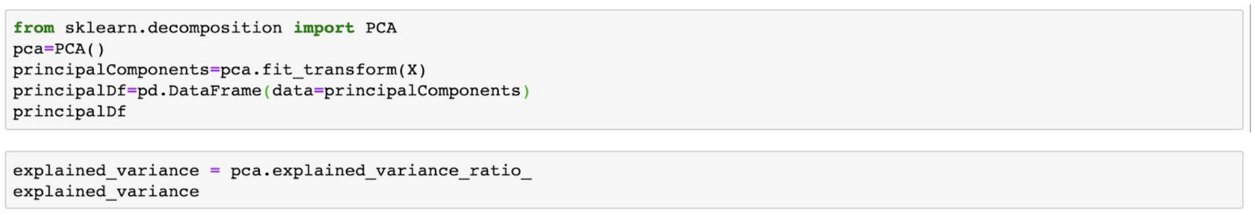
```
Accuracy: 0.7581159045896839
```

Even though the result might not be as high as 90% or more, which of course is more desirable, the fact that initially the number of features was significantly decreased might provide some explanation to this medium to high accuracy. Perhaps a feature iteration of these models in which one of these left out features is included might provide a better accuracy for this data.

Clustering:

Unsupervised learning was also performed in the following manner:

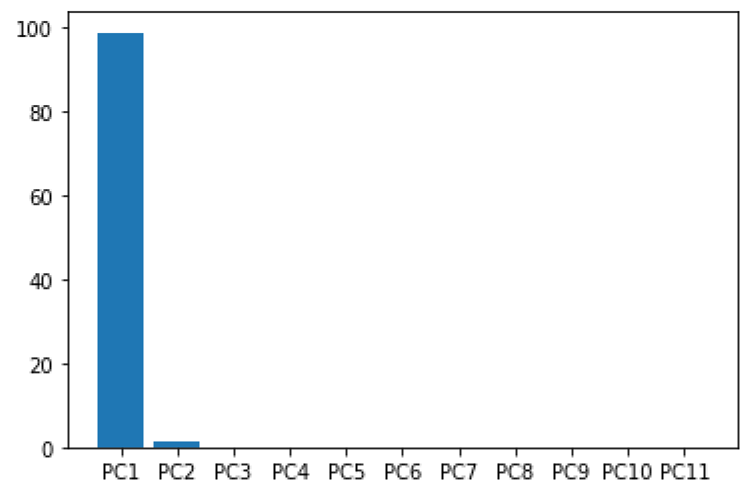
The lending club dataset is huge with around 34 descriptive features we perform Clustering techniques like PCA to reduce the dimensionality. Sklearn.decomposition provides a function for implementing PCA. Before implementing PCA all the categorical data is converted into numerical data using one hot encoding technique and normalization must be implemented on the dataset.



```
from sklearn.decomposition import PCA
pca=PCA()
principalComponents=pca.fit_transform(X)
principalDf=pd.DataFrame(data=principalComponents)
principalDf
```

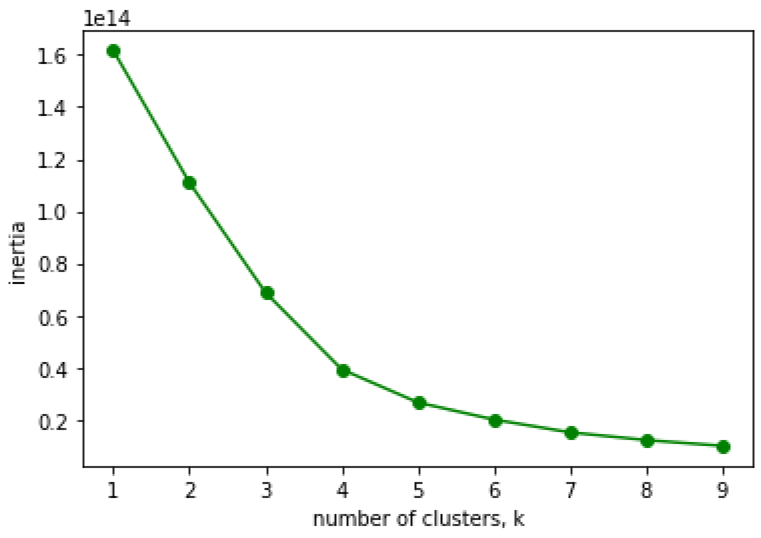
```
explained_variance = pca.explained_variance_ratio_
explained_variance
```

Explained_variance provides the variance between individual Principal Components. Using the variance we can determine how many components are required to describe the data. From the result we observe that the first component contributes nearly 90% of the data. This is a very good outcome, because the clusters are not something that can be easily interpreted, particularly on data projected by dimensionality reduction techniques such as PCA. The clusters might give good accuracy and might be in sync with the model and the data, but the lack of interpretability is a latent challenge that can be compounded by the fact that we can only visualize these groups up to 3 dimensions in a 3D plot, or better in a 2D plot.



K-Means Clustering:

To perform the k-means clustering we consider the top principal components and fir it to K-means algorithm and determine the best number of clusters.The number of clusters is determined from the inertia graph.



From the graph we observe that after K=3 the change in k value is not much significant.So we consider the value of k as 3 and perform K-means algorithm. A value of K = 3 could have some relation to the grades that were encoded, but there is no guarantee of this, again the interpretability is the hardest part of it and requires an immense amount of domain knowledge. Attempts were made to interpret these by plotting two different original features encoding each point by their predicted cluster to see if patterns emerge. No significant pattern was observed.

Finally the evaluation by using K-folds was not performed because of the big amount of data that was used, so it wasn’t a requirement at this point.