Toggle code

drawing

Week 9 - Tree-Based Methods

Dr. David Elliott

1. Imballanced Data

2. Feature Interpretation/Reduction

3. Strengths and Limitations

Dataset Example: Lending Club[18]

LendingClub (was) the world's largest peer-to-peer lending platform.

Investors were able to search and browse the loan listings on LendingClub website and select loans that they want to invest in based on the information supplied about the borrower, amount of loan, loan grade, and loan purpose.

- Investors make money from the interest on these loans.
- LendingClub makes money by charging borrowers an origination fee and investors a service fee.

Notes

- It was the first peer-to-peer lender to register its offerings as securities with the Securities and Exchange Commission (SEC), and to offer loan trading on a secondary market.
- headquartered in San Francisco, California
- It now takes an institutional investor-focused approach

- Its loan trading platform was closed down in 2020

	loan_amnt	int_rate	annual_inc	dti	total_acc	term	home_ownership	emp_length	open_acc	pub_rec	...	mort_acc	avg_cur_bal	delinq_amnt	fico_ra
0	17000.0	8.99%	105000.0	14.82	13.0	36 months	RENT	3 years	7.0	0.0	...	0.0	5321.0	0.0	
1	7500.0	15.61%	75000.0	24.14	19.0	36 months	RENT	6 years	14.0	0.0	...	3.0	17895.0	0.0	
2	12000.0	10.49%	44000.0	17.76	24.0	60 months	RENT	2 years	13.0	0.0	...	0.0	1280.0	0.0	
3	10000.0	11.05%	72000.0	8.45	20.0	36 months	MORTGAGE	5 years	6.0	0.0	...	2.0	9165.0	0.0	
4	24000.0	5.32%	235000.0	9.23	21.0	36 months	MORTGAGE	< 1 year	12.0	0.0	...	2.0	20279.0	0.0	

5 rows × 21 columns

When a person applies for a loan, there are two types of decisions that could be taken by the company[18]:

- **Loan accepted**: If the company approves the loan, there are 3 possible scenarios described below:
  - *Fully paid*: Applicant has fully paid the loan (the principal and the interest rate)
  - *Current*: Applicant is in the process of paying the instalments, i.e. the tenure of the loan is not yet completed.
  - *Charged-off*: Applicant has not paid the instalments in due time for a long period of time, i.e. he/she has defaulted on the loan
- **Loan rejected**: The company had rejected the loan (because the candidate does not meet their requirements etc.).

Notes

- *"It now presents the algorithm just as a search tool for investors to find Notes they would like to purchase, using borrower and loan attributes such as the length of a loan term, target weighted average interest rate, borrower credit score, employment tenure, homeownership status, and others.[63]"*
- *"To reduce default risk, LendingClub focuses on high-credit-worthy borrowers, declining approximately 90% of the loan applications it received as of 2012[64] and assigning higher interest rates to riskier borrowers within its credit criteria.[23]"*
- *"Only borrowers with FICO score of 660 or higher can be approved for loans.[54]"*
- Lending Club blends traditional credit reports with data gathered from around the web.

```
660.0
```

When the company receives a loan application, the company has to make a decision for loan approval based on the applicant's profile. Two types of risks are associated with the bank's decision:

- If the applicant is likely to repay the loan, then not approving the loan results in a loss of business to the company
- If the applicant is not likely to repay the loan, i.e. he/she is likely to default, then approving the loan may lead to a financial loss for the company

The data given contains the information about past loan applicants and whether they 'defaulted' or not. The aim is to identify patterns which indicate if a person is likely to default, which may be used for taking actions such as denying the loan, reducing the amount of loan, lending (to risky applicants) at a higher interest rate, etc.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16775 entries, 0 to 16774
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   loan_amnt             16775 non-null  float64
 1   int_rate              16775 non-null  object 
 2   annual_inc            16775 non-null  float64
 3   dti                   16775 non-null  float64
 4   total_acc             16775 non-null  float64
 5   term                  16775 non-null  object 
 6   home_ownership        16775 non-null  object 
 7   emp_length            16775 non-null  object 
 8   open_acc              16775 non-null  float64
 9   pub_rec               16775 non-null  float64
 10  pub_rec_bankruptcies  16775 non-null  float64
 11  mort_acc              16775 non-null  float64
 12  avg_cur_bal           16775 non-null  float64
 13  delinq_amnt           16775 non-null  float64
 14  fico_range_high       16775 non-null  float64
 15  fico_range_low        16775 non-null  float64
 16  num_bc_tl             16775 non-null  float64
 17  num_tl_90g_dpd_24m    16775 non-null  float64
 18  zip_code              16775 non-null  object 
 19  installment           16775 non-null  float64
 20  loan_status           16775 non-null  object 
```

```
dtypes: float64(15), object(6)
memory usage: 2.7+ MB
```

	Description
annual_inc	The self-reported annual income provided by the borrower during registration.
avg_cur_bal	Average current balance of all accounts
delinq_amnt	The past-due amount owed for the accounts on which the borrower is now delinquent.
dti	A ratio calculated using the borrower’s total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower’s self-reported monthly income.
emp_length	Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
fico_range_high	The upper boundary range the borrower’s FICO at loan origination belongs to.
fico_range_low	The lower boundary range the borrower’s FICO at loan origination belongs to.
home_ownership	The home ownership status provided by the borrower during registration or obtained from the credit report. Our values are: RENT, OWN, MORTGAGE, OTHER
installment	The monthly payment owed by the borrower if the loan originates.
int_rate	Interest Rate on the loan
loan_amnt	The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
loan_status	Current status of the loan
mort_acc	Number of mortgage accounts.
num_bc_tl	Number of bankcard accounts
num_tl_90g_dpd_24m	Number of accounts 90 or more days past due in last 24 months
open_acc	The number of open credit lines in the borrower's credit file.
pub_rec	Number of derogatory public records
pub_rec_bankruptcies	Number of public record bankruptcies
term	The number of payments on the loan. Values are in months and can be either 36 or 60.
total_acc	The total number of credit lines currently in the borrower's credit file
zip_code	The first 3 numbers of the zip code provided by the borrower in the loan application.

```
As the xlwt package is no longer maintained, the xlwt engine will be removed in a future version of pandas. This is the only en
gine in pandas that supports writing in the xls format. Install openpyxl and write to an xlsx file instead.

: boolean
    use_inf_as_null had been deprecated and will be removed in a future
    version. Use `use_inf_as_na` instead.
```

Loan Status (%)

```
Fully Paid     80.81
Charged Off    19.19
Name: loan_status, dtype: float64
```

Data Prep

For the purposes of this lecture, we are only using a sample of the full dataset (~1%). This is to ensure things dont take an age to run.

- You can have a look at how I prepared the data and reduced it down in the "Loan_Club_Explore_and_Prep.ipynb" found in the "Extra" folder.
- You could also try play with the larger sample of data.
- There are also 150 columns so loads of features, some of them potentially concerning!

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16775 entries, 0 to 16774
Data columns (total 19 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   loan_amnt             16775 non-null  float64
 1   int_rate              16775 non-null  float64
 2   annual_inc            16775 non-null  float64
 3   dti                   16775 non-null  float64
 4   total_acc             16775 non-null  float64
 5   term                  16775 non-null  int32  
 6   emp_length            16775 non-null  int32  
 7   open_acc              16775 non-null  float64
 8   pub_rec               16775 non-null  float64
 9   pub_rec_bankruptcies  16775 non-null  float64
 10  mort_acc              16775 non-null  float64
 11  avg_cur_bal           16775 non-null  float64
 12  delinq_amnt           16775 non-null  float64
 13  fico_range_high       16775 non-null  float64
 14  fico_range_low        16775 non-null  float64
 15  num_bc_tl             16775 non-null  float64
```

```
 16  num_tl_90g_dpd_24m    16775 non-null  float64
 17  installment           16775 non-null  float64
 18  loan_status           16775 non-null  object
dtypes: float64(16), int32(2), object(1)
memory usage: 2.3+ MB
```

7. Imballanced Data

Forest

We can deal with class imballance using `class_weight = 'balanced'`, alike to discussed last week.

We can also undersample using a *ballanced random forest*, which builds several estimators on different randomly selected subsets of data.

Generally what performs better depends on the amount of data you are training on and the aims of the model.

Notes

- Balanced Random Forests are generally faster to train.
- If data is small then class weight will probably be better (as seen below), but if you have very large datasets, then undersampling will likely work better.

```
Random Forest
Elapsed time: 0.68 seconds
OOB Score: 0.803
Random Forest (Balanced Class Weight)
Elapsed time: 0.73 seconds
OOB Score: 0.803
Balanced Random Forest
Elapsed time: 0.64 seconds
OOB Score: 0.635
```

Notes

- Remember from last week, accuracy can only tell us so much!

Figure 1: Random Forest Validation Confusion Matrix

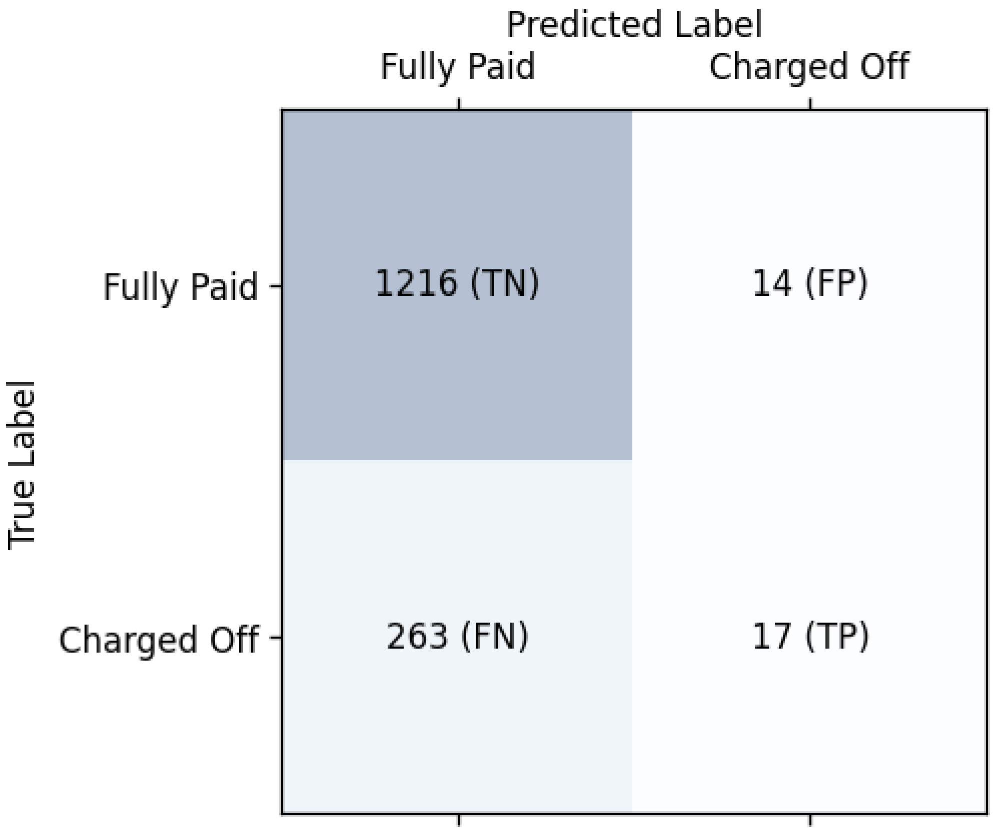


Figure 2: Random Forest (Balanced Class Weight) Validation Confusion Matrix

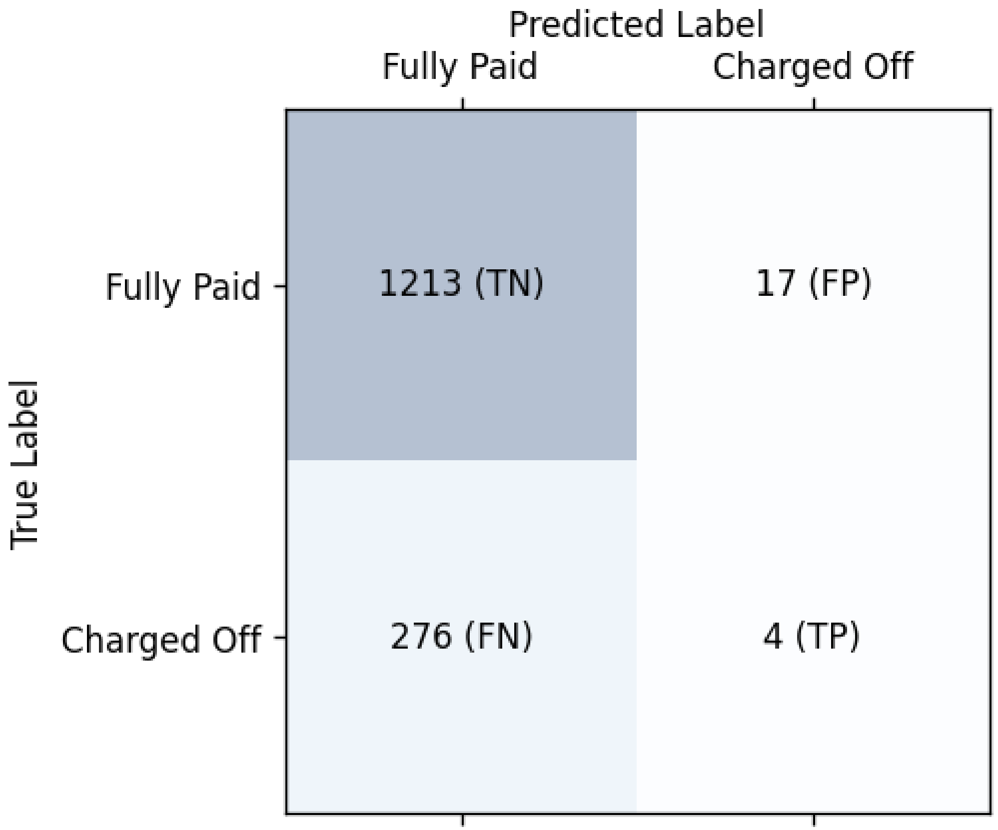
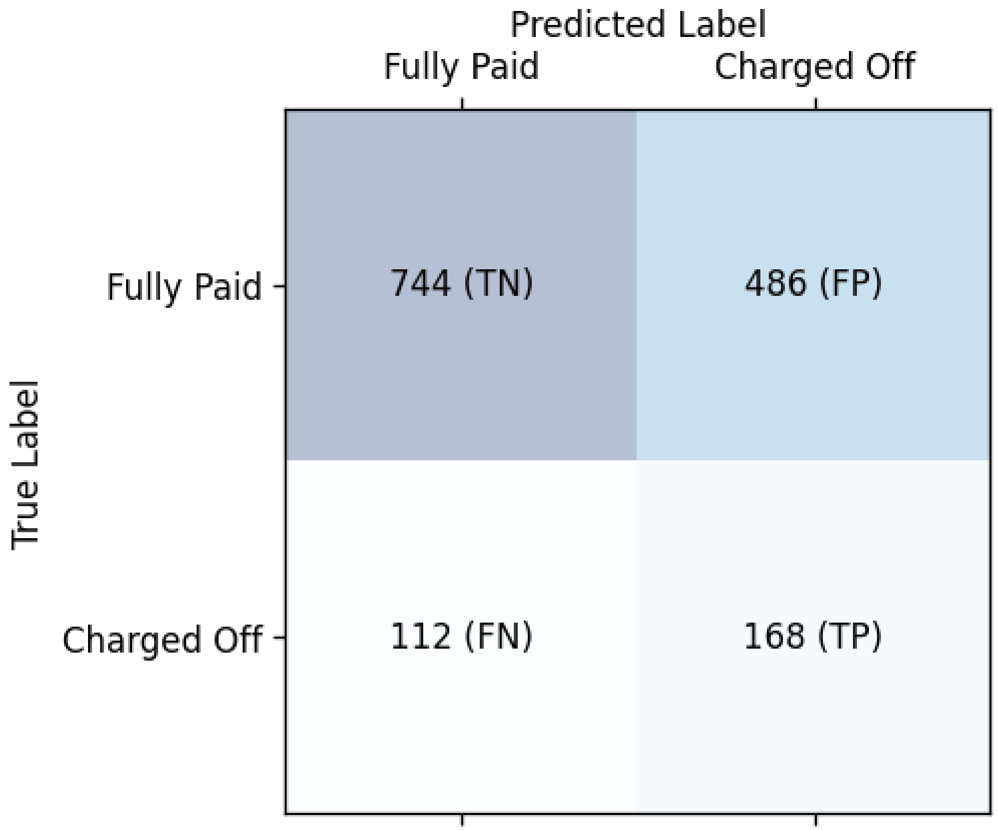


Figure 3: Balanced Random Forest Validation Confusion Matrix



Notes

- This looks like the problem we had last week, depending on what we try we either have poor performance on the minority class ("Charged Off") or have a high false positive rate.

classifer	metric	Fully Paid	Charged Off	accuracy	macro avg	weighted avg
Random Forest	**precision**	0.82	0.55	0.82	0.69	0.77

classifer	metric	Fully Paid	Charged Off	accuracy	macro avg	weighted avg
	recall	0.99	0.06	0.82	0.52	0.82
	f1-score	0.90	0.11	0.82	0.50	0.75
	support	1230.00	280.00	0.82	1510.00	1510.00
Random Forest (Balanced Class Weight)	precision	0.81	0.19	0.81	0.50	0.70
	recall	0.99	0.01	0.81	0.50	0.81
	f1-score	0.89	0.03	0.81	0.46	0.73
	support	1230.00	280.00	0.81	1510.00	1510.00
Balanced Random Forest	precision	0.87	0.26	0.60	0.56	0.76
	recall	0.60	0.60	0.60	0.60	0.60
	f1-score	0.71	0.36	0.60	0.54	0.65
	support	1230.00	280.00	0.60	1510.00	1510.00

The above was just done on pretty much the default params, so you'd want to do some searches to get better hyperparameters.

	param_n_estimators	param_max_features	param_class_weight	mean_test_score	std_test_score
28	105	11	None	0.103820	0.016114
25	151	12	None	0.100164	0.012414
39	97	7	None	0.097164	0.012689
4	291	12	None	0.096904	0.013131
0	120	17	None	0.095412	0.007575

	param_n_estimators	param_max_features	mean_test_score	std_test_score
30	334	3	0.407487	0.015900
43	125	3	0.406370	0.017198

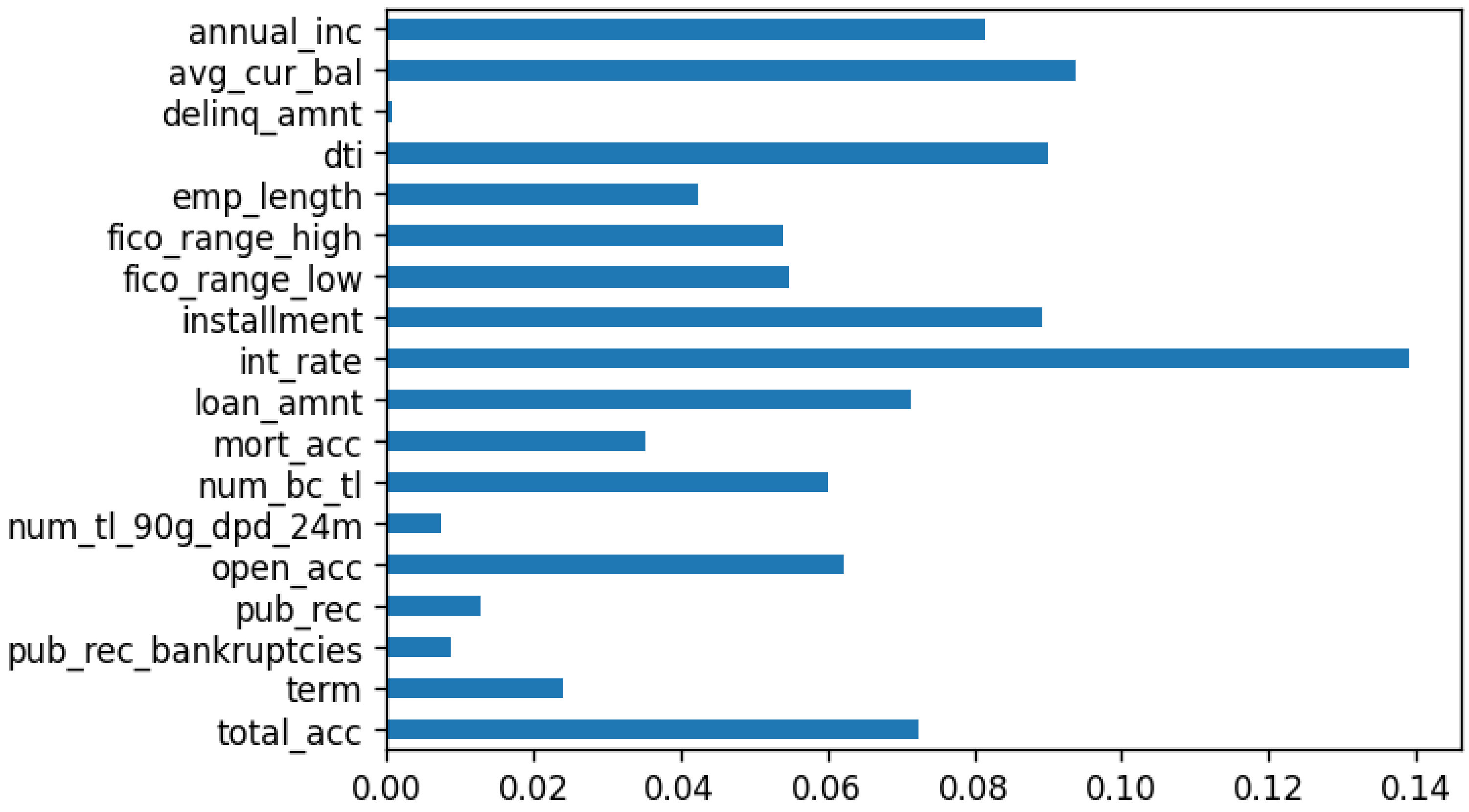
	param_n_estimators	param_max_features	mean_test_score	std_test_score
51	417	3	0.405944	0.017373
19	385	4	0.405593	0.013208
55	456	8	0.404155	0.017343

8. Feature Interpretation/Reduction

Feature importances

Feature importances, as we have previously seen, can give us insight into the important features for our model.

Figure 4: Average Feature Importances for a Balanced Random Forest



We could also use the importance of each feature (using average impurity decrease), for feature selection.

We can put it in a pipeline and use the `SelectFromModel` function from Scikit-learn.

Notes

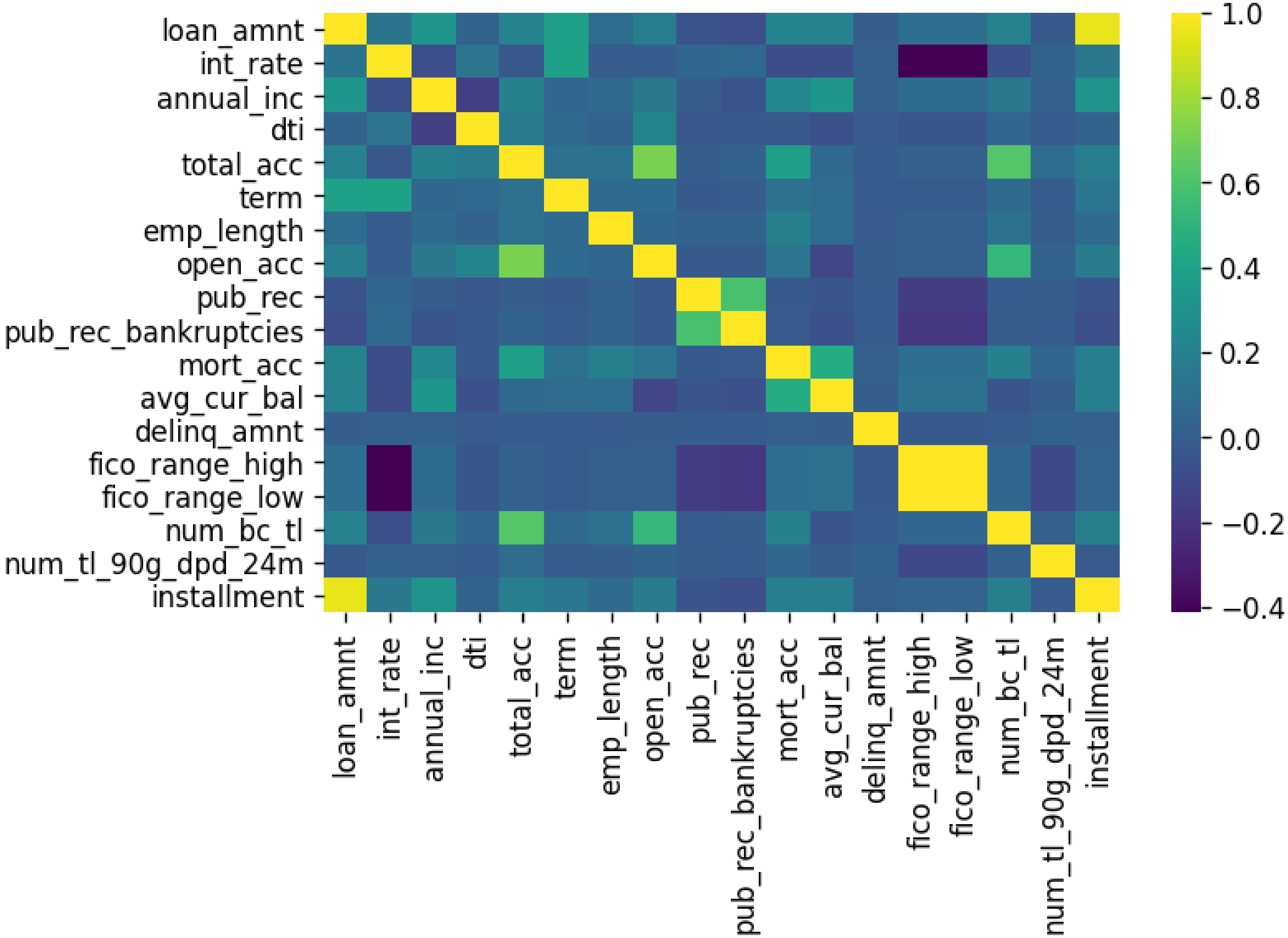
- We can provide both a numeric theshold or use a heuristic such as the mean and median[5].

```
SVM
CV f1-score: 0.343 +/- 0.007
```

```
Forest SVM
CV f1-score: 0.342 +/- 0.006
```

Limitations

- Feature importances can be misleading for high cardinality features (many unique values)[6]
- If features are highly correlated, one feature may be ranked highly while the information of the others not being fully captured[4].



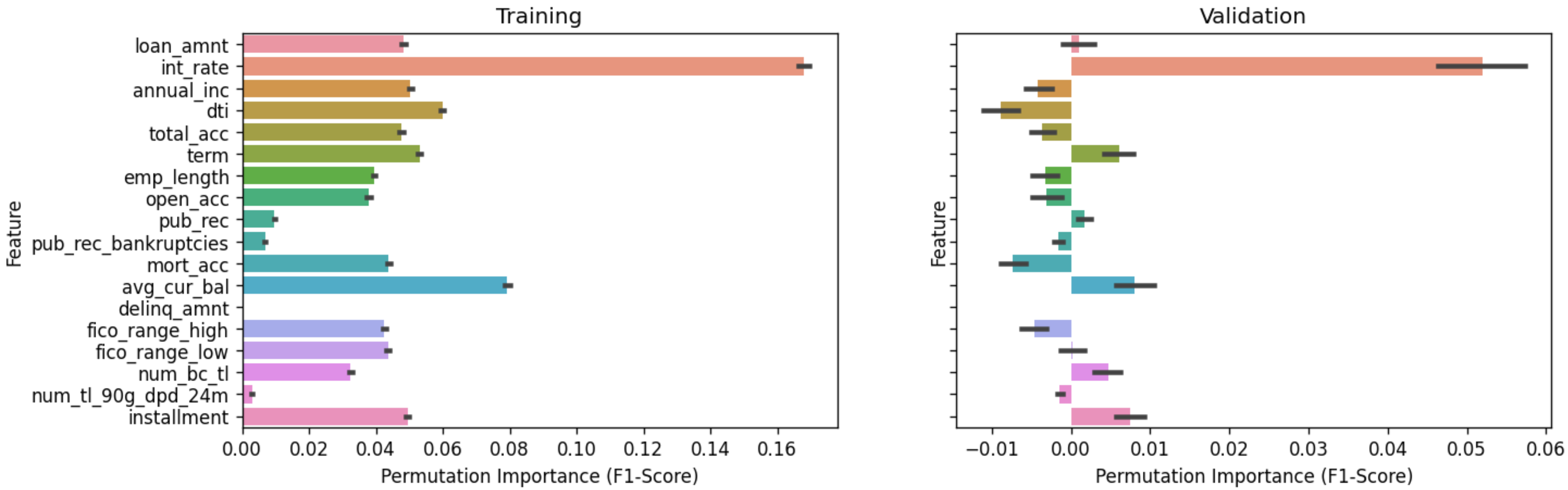
Permutation Importance

The permutation feature importance is defined to be the decrease in a model score when a single feature value is randomly shuffled[7].

This procedure breaks the relationship between the feature and the target, thus the drop in the model score is indicative of how much the model depends on the feature[8].

Notes

- *" In the permutation-based approach, for each tree, the OOB sample is passed down the tree and the prediction accuracy is recorded. Then the values for each variable (one at a time) are randomly permuted and the accuracy is again computed. The decrease in accuracy as a result of this randomly shuffling of feature values is averaged over all the trees for each predictor. The variables with the largest average decrease in accuracy are considered most important."*[9]
- *"This technique benefits from being model agnostic and can be calculated many times with different permutations of the feature."* https://scikit-learn.org/stable/modules/permutation_importance.html
- *"Using a held-out set makes it possible to highlight which features contribute the most to the generalization power of the inspected model. Features that are important on the training set but not on the held-out set might cause the model to overfit."* https://scikit-learn.org/stable/modules/permutation_importance.html
- *"Warning: Features that are deemed of low importance for a bad model (low cross-validation score) could be very important for a good model. Therefore it is always important to evaluate the predictive power of a model using a held-out set (or better with cross-validation) prior to computing importances. Permutation importance does not reflect to the intrinsic predictive value of a feature by itself but how important this feature is for a particular model."* https://scikit-learn.org/stable/modules/permutation_importance.html
- For more examples of methods of interpreting forests see: https://towardsdatascience.com/explaining-feature-importance-by-example-of-a-random-forest-d9166011959e



Extra: Fairness

In last weeks "Applications" notes I questioned what information is fair for us to use when making decisions around lending. Well this week we have a dataset which has some of these very varibles in which could be problematic.

- zip_code
  - Maybe you think people who live in wealthy areas are more likely to pay back the loan?
  - Doesn't this make it worse for people financially responcible people who live in poor areas?
  - Might this unfortunately be related to someones race/ethnicity?
- pub_rec (Number of derogatory public records)
  - What do they define as a "derogatory public record"?
  - Should we use such a vauge feature that could include discrimatory information?
  - Does this include criminal history? Is the american criminal system unbias? Is this likely to account for certain types of crime over white colar crime?
- homeownership

    - Among racial demographics in America, white people have the country's highest homeownership rate, while African Americans have the lowest.
    - One study shows that homeownership rates appear correlated with higher school attainment. ("A Note on the Benefits of Homeownership, Federal Reserve Bank of Chicago" (PDF). Chicagofed.org. Retrieved October 14, 2017.)

If you look at the full list of varibles for this dataset (we've used a subset), you'll see there are even more features of which you may question their validity or fairness. Should we be including these varibles in our models, in other words should we be assigning weights to them? Without other demographic information (race, gender, ect.) can we ever try correct for them in our model?

More on this in week 11...

Note

- Lending Club use what Cathy O'Neil terms e-scores. You can read more about the issues with e-scores in chapter 8 of Weapons of Math Destruction.

9. Strengths and Limitations

There are always advantages and disadvantages to using any model on a particular dataset.

Trees

Advantages[1,3,10]

- Easy to explain
    - Trees can be displayed graphically in an interpretable mannor.
- Make few assumptions about the training data (non-parametric)
    - e.g. we don't assume the data is linear.
- Inherently multiclass
    - Can also handle multitask output (multiclass-multioutput)
- Can handle different types of predictors*

    - Independent of feature scaling
- Can handle missing values*
- Can handle multitask output (multiclass-multioutput)

* limited or unavailable in Scikit-Learn

Notes

- *"Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic."*[12]
- Decision trees potentially more mirror human decision-making than the regression and classification approaches previously discussed.
- Generally, you'll find tree impliminations that are specialised for particular types of data.
- *"Warning At present, no metric in sklearn.metrics supports the multiclass-multioutput classification task."*[11]

Extra: Categorical Features and Sklearn

You may also be wondering: where are my previous data visualisations of the categorical data before this? Well Sklearn's CART decision trees currently *"does not support categorical variables"*. This means:

- Do not use `Label Encoding` if your categorical data is **not ordinal** with `DecisionTreeClassifier()`, you'll end up with splits that do not make sense, as the data will be treat as numeric[13].

- Using a `OneHotEncoder` is the only current valid way with sklearn, allowing arbitrary splits not dependent on the label ordering, but is computationally expensive and it can deteriorate the performance of decision trees as it leads to sparse features, which can mess up feature importance[12].

Solutions

Currently the best way of handling categorical features is to use `h2o.randomForest`, a different forest implimentation, or `catboost`, a boosting classifier.

Notes

- H2o has sklearn support which may be useful to look into: http://docs.h2o.ai/h2o/latest-stable/h2o-docs/h2o-client.html

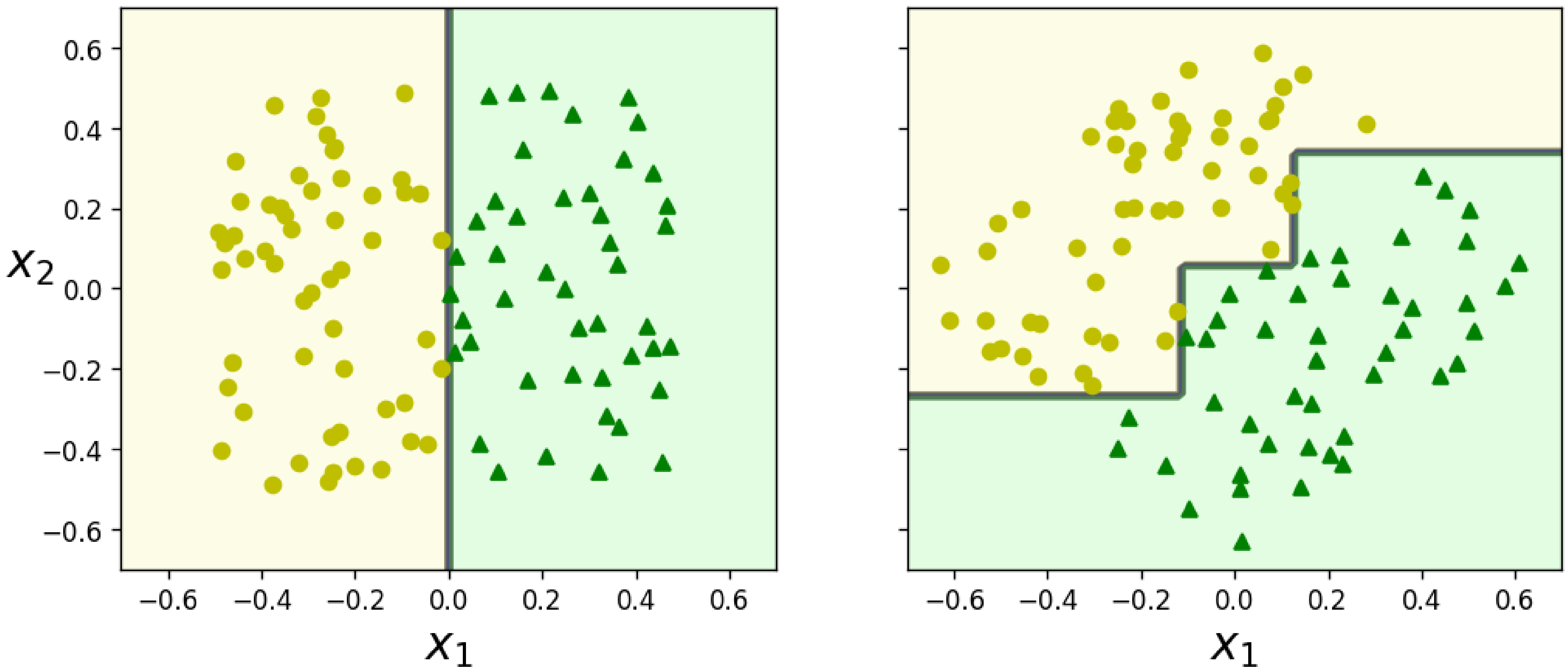
Disadvantages[1,10]

- Comparatively poor generalization performance.
- Easy to overfit
  - Require pruning
- High variance
  - A small change in the data can cause a large change in the estimated tree.
- Orthogonal decision boundaries
  - Model affected by the rotation of the data.
- Cannot guarantee to return the globally optimal decision tree.
  - locally optimal decisions are made at each node.

Notes

- If there is a highly non-linear and complex relationship between the features and the response then decision trees may outperform classical approaches. However if the relationship between the features and the response is well approximated by a linear model, then an approach such as linear regression will likely work well[1].
- A decision tree is quite boxy. How the model makes a decision boundary is going to be affected by the rotation of the data (as DTs create straight lines).

Extra: Sensitivity to Rotation



Forests/ExtraTrees

Generally improve upon trees, at the expense of interpretability.

Advantages

- Comparatively good generalisability

    - Not too affected by outlier observations.
    - Can still overfit[16] but much less likely
- Comparatively small variability in prediction accuracy when tuning[14]
- Comparatively good “out of the box” performance[15]
    - Easy of use
    - not much tuning required to get good results
- ExtraTrees is faster to train than random forests
    - Its time consuming to to find the best theshold for each feature at each node[3].
- Work well on large datasets

Disadvantages

- Not good for very high-dimensional sparse data[17].
    - e.g. text
- Harder than trees to interpret
    - Trees in random forests tend to be deeper than decision trees (due to feature subsets)[17].
- More computationally demanding to train than other algorithms[17]
    - Require more memory and are slower to train and to predict than linear models.
    - However, they can be easily parallelized across multiple CPU cores.
- Setting different random states can drastically change the model[17].
    - The more trees, the more robust to this they are to this.

Notes

- Using more CPU cores will result in linear speed-ups[17].

- _'As concluding remarks about ensemble techniques, it is worth noting that ensemble learning increases the computational complexity compared to individual classifiers. In practice, we need to think carefully about whether we want to pay the price of increased computational costs for an often relatively modest improvement in predictive performance. An often-cited example of this tradeoff is the famous $1 million Netflix Prize, which was won using ensemble techniques. The details about the algorithm were published in The BigChaos Solution to the Netflix Grand Prize by A. Toescher, M. Jahrer, and R. M. Bell, Netflix Prize documentation, 2009, which is available at http://www.stat.osu.edu/~dmsl/GrandPrize2009_BPC_BigChaos.pdf. The winning team received the $1 million grand prize money; however, Netflix never implemented their model due to its complexity, which made it infeasible for a real-world application: "We evaluated some of the new methods offline but the additional accuracy gains that we measured did not seem to justify the engineering effort needed to bring them into a production environment." http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html'_ [4]

References

1. James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An introduction to statistical learning. Vol. 112. New York: springer, 2013.
2. Gorman KB, Williams TD, Fraser WR (2014). Ecological sexual dimorphism and environmental variability within a community of Antarctic penguins (genus Pygoscelis). PLoS ONE 9(3):e90081. https://doi.org/10.1371/journal.pone.0090081
3. Géron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. " O'Reilly Media, Inc.".
4. Raschka, Sebastian, and Vahid Mirjalili. Python Machine Learning, 2nd Ed. Packt Publishing, 2017.
5. http://scikit-learn.org/stable/modules/feature_selection.html
6. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier
7. L. Breiman, “Random Forests”, Machine Learning, 45(1), 5-32, 2001
8. https://scikit-learn.org/stable/modules/permutation_importance.html
9. https://bradleyboehmke.github.io/HOML/random-forest.html
10. https://github.com/rasbt/stat479-machine-learning-fs19/blob/master/06_trees/06-trees__notes.pdf
11. https://scikit-learn.org/stable/modules/multiclass.html
12. https://scikit-learn.org/stable/modules/tree.html
13. https://stackoverflow.com/questions/38108832/passing-categorical-data-to-sklearn-decision-tree
14. Probst, Philipp, Bernd Bischl, and Anne-Laure Boulesteix. 2018. “Tunability: Importance of Hyperparameters of Machine Learning Algorithms.” arXiv Preprint arXiv:1802.09596.
15. https://github.com/rasbt/stat451-machine-learning-fs20/blob/master/L07/07-ensembles__notes.pdf.

16. Segal MR: Machine Learning Benchmarks and Random Forest Regression. Technical Report, Center for Bioinformatics & Molecular Biostatistics, University of California, San Francisco 2004.
17. Müller, A. C., & Guido, S. (2016). Introduction to machine learning with Python: a guide for data scientists. " O'Reilly Media, Inc.".
18. https://www.kaggle.com/faressayah/lending-club-loan-defaulters-prediction