Capstone project:
Credit risk on hire purchases
Machine Learning Engineer Nanodegree

Stefano Ernesto Bourscheid
Udacity
29/10/2017

I. Definition

Project Overview

In Brazil exists a very common and traditional method of payment: the hire purchases. Hire purchases work basically as a money lending, in which the shopkeeper let the customer buy an amount of products in exchange of a payment promise, with defined payment dates and values usually followed by a 'contract' that the customer need to sign.

Right now the only protection that the shopkeeper have against payment failures is that he is allowed to 'blacklist' the customer in public databases, potentially preventing that person to perform more hire purchases in other places until they debt is paid - or a time limit is reached.

The problem is that this system has only negative informations, driving Brazil into an even more critical economical situation: according to Serasa Experian in 2016, 30% of the buying population is blacklisted in their database. This issue drives Brazilians that are struggling with financial issues into an even worse condition, while preventing sales from happening.

The problem that I will try to solve in this project is to use data science as a tool to generate better credit analysis, working with a Brazilian startup named MeuCrediario that aims to solve this problem connecting independent stores to profile customers and their behaviors on hire purchases.

Problem Statement

Working with an anonymous dataset from MeuCrediario that contains informations about past contracts and their respective customers information, the goal is to create a model that predicts how many days the customer will delay the payment for the proposed contract. We can break down this process by the following steps:

1. The shopkeeper inputs the customer personal data in MeuCrediario’s application.
2. The shopkeeper inputs the proposed contract information in MeuCrediario’s application.
3. The model (running in the cloud) will analyze the information and give back a ‘score’ to that contract.
4. The shopkeeper will use the information received to make the decision of accepting of denying the proposed contract.

This project will provide only the model that MeuCrediario would use to make the credit analysis. We can break down this task in the following steps:

1. Download a dataset from MeuCrediario’s past hire purchases;
2. Preprocess the data removing noise and unnecessary features;
3. Train the model using the processed features;

Metrics

The method to measure the model efficiency is the **Coefficient of Determination** (or R squared), given that the model being trained is a regression and is easier to understand than RMSE for example (since it always go between 0 and 1). Also, it agrees perfectly with the further observations and evaluations on the results.
It may be described as:

$$R^2 = 1 - \frac{SSres}{SStot}$$

The end result is a number ranging from -1 to 1, demonstrating how much variance is our model describing in relationship with the dataset (more is better).

Breaking down this formula, we have:

- *SSres* is the residual sum of squares, defined as:

$$\mathrm{ss}_{\mathrm{Res}} = \sum_{\mathrm{i}} (\mathrm{y_i} - \mathrm{f_i})^2$$

In the above representation, *y* is the correct value or *label* and *f* is the predicted value.

- *SStot* may be defined as:

$$SS_{TOT} = \sum_{i} (y_i - \bar{y})^2$$

Finally, $\bar{y}$ is the mean of the labels:

$$\bar{y} = \frac{1}{N} \sum_{i=0}^{N} y_i$$

II. Analysis

Data Exploration

The dataset being used in this project has been provided by MeuCrediario and is a sample of 2500 sales (or contracts) done in the last 4 years (since 2013).

Here I will demonstrate an example of the dataset and all its features, along with some discussions and curiosities about it:

Column name	**Number of not-nulls**	**Sample value**	**Description**
faturado	2500	114.90	Total value of the contract
entrada	0	(null)	If the customer made an 'entry', which is a payment in advance done in the first day
qtd_parcelas	2500	4	Number of parcels that the contract is divided
valor_financiado	2500	114.90	Total value minus the 'entry' if, applicable
valor_parcela	2500	28.73	Total value divided by the number or parcels
qtd_consultas_ate15dias	1259	0	Number of times that the customer was **searched** in the Brazilian SPC (aka blacklist) database in the last 15 days
qtd_consultas_ate60dias	1259	0	Number of times that the customer was **searched** in the Brazilian SPC (aka blacklist) database in the last 60 days

qtd_registros	1259	0	Number of active occurrences registered in the SPC database (aka blacklist)
qtd_contratos_quitados_cliente	2500	7	Number of past paid contracts of this customer
qtd_contratos_abertos_cliente	1420	(null)	Number of open active contracts of this customer
media_atraso_cliente	2266	0	Mean of the days that the customer delayed payments in the last contracts
sexo	2500	2	Sex, 1=man / 2=woman
idade	2500	66	Age in years
tel_fixo	2500	True	If this customer currently have fixed telephone in his home
estado_civil	2500	1	The marital status of the customer: 1=single, 2=married, 3=divorced, 4=widow, 5=other
dependentes	1174	(null)	Number of dependents within the family
veiculo	1174	(null)	The main transport type: 1=car, 2=motorcycle
veiculo_financiado	1016	(null)	If the customer have a financed car/ motorcycle
moradia	1175	(null)	Main situation of the customer home: 1=rent, 2=owned, 3=live with the parents
ocupacao	2498	24	An integer representing one of the 574 occupations recognized by Brazilian government. 24 for an example is house wife
telefone_comercial	2500	True	If the customer have a commercial contact (phone) at his job/organization
anos_empresa	2500	24	Number of years working on the same job and company
salario	2481	800	Monthly salary in Brazilian reais (R$)
mda_cliente	2500	3	In his payment history, the biggest number of days that the customer ever delayed a payment

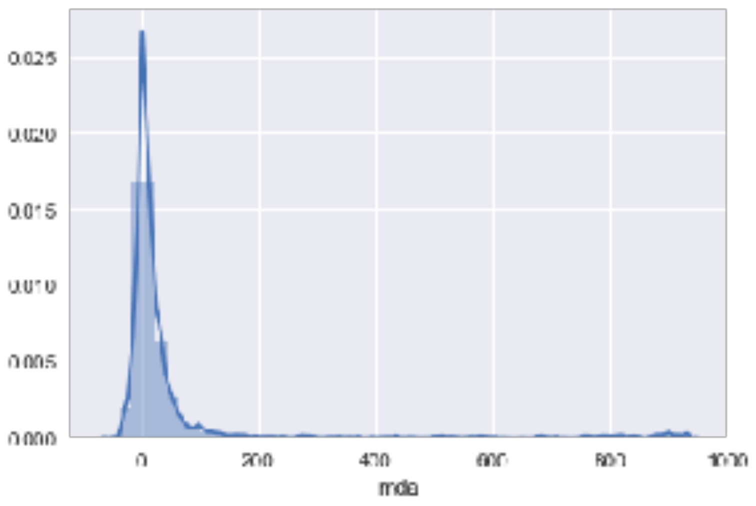
Along these 25 features, the label column is `mda`, which can be described as the biggest number of days that the customer delayed a payment in this contract. This number may be negative, if the customer always paid in advance.

Some issues that we can notice is the number of features - 25 - and also the huge categorical number of occupations, which will be addressed furthermore.

Exploratory Visualization

The first thing that we will analyze is a distribution plot about the label to see what is our output space and if the distribution is gaussian.

Distribution plot on column ['mda']



With this chart we can conclude that even though the range goes up to 1000, we can argue that only roughly 5% are above 200.

Algorithms & Techniques

To cluster the occupations I used **K-Modes**, which is basically a K-Means algorithm but instead of means it used modes to describe the distance metric between the occupations. This was necessary because there is no sense of distance between the occupations, for example: is a secretary closer to a shopkeeper than a salesman? The answer to this question was necessary because K-Means requires a sense of distance between the entities.

The solution was an extension to the original K-Means made by Zhexue Huang, in 1998. In Huang's own words: "[…] working only on numeric values prohibits it from being used to cluster real world data containing categorical values.". K-Modes allow us to cluster categorical data by the frequency that they occur, which is exactly what we need.

In the K-modes clustering algorithm we need to defined a number of clusters that it will extract from the data. The first thing that it does is to give an initial solution at random with the vectors resulted from the means of the `mda` feature and the mode of the occupations, classifying each input to the closest cluster according the the previous classification. Until a threshold is passed (convergence), this process repeats classifying the inputs.

To predict the new labels I've used the Random Forest Regressor, which is a supervised learning ensemble technique in which several decision trees select randomly a limited amount of features per tree, and they 'vote' their prediction given the features selected.

This model is specially good with a high amount of features and in complex problems, even though the model itself is not simple (is quite a blackbox) it can be tuned and usually have very good scores with real world problems.
To choose the best tuning for the model I've used the GridSearch by sklearn, in which it tests all the possibilites (given by me) of combinations with the parameters, keeping the ones with the highest score. This saves a lot of time in trial-and-error specially when dealing with complex problems that are hard to solve and have a mathematical sense of what's going to be improved.

Benchmarks

To benchmark the model, I've chosen a very simple algorithm to test the solution with: Decision Trees Regressor.

After training it with the training data, the R2 scoring pointed out a very very high overfitting issue with this initial model:

Training data: 1.0
Testing data: 0.0

Even though the testing score is very high, is hard to believe that this overfitting issue would go away with more data, therefore the goal of the final modal is to beat this simple one with less overfitting.

III. Methodology

Data Preprocessing

As stated before on the Data Exploration section, we have outliers in the label. In order to solve this problem, the dataset was queried to remove the lines in which the ‘mda’ was higher than 200, keeping almost 94% of the data - but reducing the range by 80%.

Another step that was done into preprocessing the data is filling all the nulls in the dataset with zeros. This is possible because every feature is of type integer and zeros have no meaning in categorical or numerical data.

Finally, the last step into preprocessing the data was to cluster the occupation categorical feature due to its huge size. This was accomplished by using a python library written according to the previously explained K-Modes technique.

Implementation

To remove all the labels higher than 200, first I checked whether this would be a good decision:

```
str(len(data.query('mda>200')) * 100 / len(data))
```

The code above uses a pandas DataFrame method called query, which filters the lines that meet the expression. In this case, the code output was *6.4*. This means that only 6.4 percent of the data have labels bigger than 200, justifying the removal of these lines in the code below:

```
features = data.query(‘mda<=200')
```

Using a feature from pandas library as well, the code below will replace any Nulls/NaNs/Nones with zeroes:

```
features = features.fillna(0)
```

And finally, the code to perform the clustering of the occupation feature imports a python library (K-Modes) to aid the implementation:

```
from kmodes import kmodes
km = kmodes.KModes(n_clusters=10, init='Huang', n_init=5,
verbose=0)
clusters = km.fit_predict(features[['ocupacao', 'mda']])
features[['ocupacao']] = clusters
```

After clustering the results, the last line of code replaces the previous feature with the results from the clustering.

Done with the preprocessing now it's time to train the simple benchmarking model in which I've chosen Decision Forest:

```
from sklearn.tree import DecisionTreeRegressor
model1 = DecisionTreeRegressor(random_state=0)

model1.fit(features_train, labels_train)
print 'Score de treino modelo benchmarking: ' +
str(model1.score(features_train, labels_train))
print 'Score de teste modelo benchmarking: ' +
str(model1.score(features_test, labels_test))
```

And finally the real model using Random Forest Regressor from which I've chosen the RandomForestRegression:

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(random_state=0)
model.fit(features_train, labels_train)

print 'Score de treino modelo final : ' +
str(model.score(features_train, labels_train))
print 'Score de teste modelo final: ' +
str(model.score(features_test, labels_test))
```

Overall, the hardest part on the project in terms of code was to make the refinement on the model (next chapter). Mainly because I couldn't select too many parameters to throw into gridsearch otherwise it would become very slow (quadratic complexity).

Refinement

As the final model, I've chosen to use GridSearch to tune the parameters of the model, trying to bring down the overfitting issue on the raw random forest algorithm:

```
param = {
    'n_estimators': [10,15,20,30],
    'criterion': ('mse', 'mae'),
    'max_features': ['sqrt','log2'],
    'min_samples_split': [2,0.3,0.5,0.7],
}
rgs = GridSearchCV(estimator=model, param_grid=param)
rgs.fit(features_train, labels_train)
```

With the tuned model the results were better indeed, both the training and the testing scores:

Training score without tuning: 0.87
Training score with tuned model: 0.90

Testing score without tuning: 0.26
Training score with tuned model: 0.31

IV. Results

Model Evaluation & Validation

The final model Random Forest Regressor was chosen due to trial and error: was the best method compared to AdaBoost and Gradient Boost for example.

The results from the evaluation were not good, because even though the training data produce a very good result (0.90), the test set doesn't (0.31). The model is clearly overfitted, and this is very probably because of the high amount of features in comparison of the small amount of data provided.

Even though the “curse of dimensionality” plays a role here, reducing the dimensionality of the data with Principal Component Analysis does not help the final result either, meaning that the dataset must be way bigger to extract all the meaning from the features.

Since the chosen model has a random factor, I’ve tried setting 5 different random states to run the scores on the tuned model:

1st run: Training: 0.90, Testing: 0.31
2nd run: Training: 0.90, Testing: 0.27
3rd run: Training: 0.90, Testing: 0.27
4th run: Training: 0.90, Testing: 0.30
5th run: Training: 0.89, Testing: 0.30

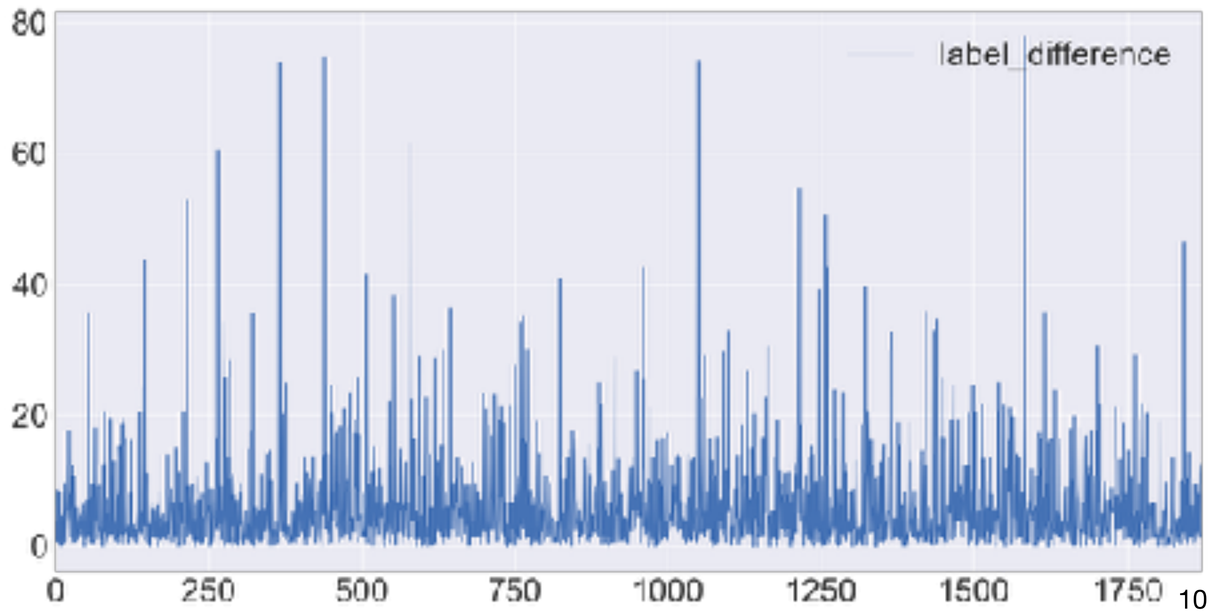
We could then use the average of these simulations as the real final score for the tuned model: **Training 0.90 and testing 0.29**.

This is clearly better and with less overfitting than the simple benchmarking model: **Training: 1.0 and testing 0.0**

Justification

To compare the results obtained from the benchmark, a graph was made to exemplify the differences between the real label vs the predicted one, first with the training dataset:

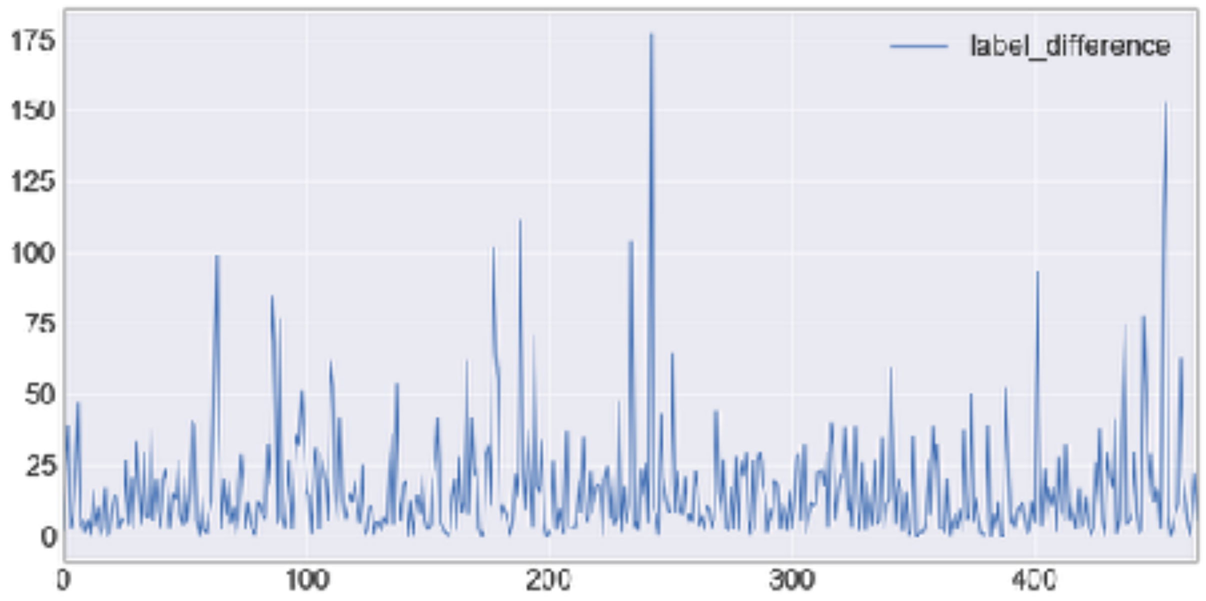
TRAINING SET - DIFFERENCES BETWEEN PREDICTED VS REAL



In the training set, we have a seemingly good result, with an average of 6.02 of difference between the predicted and the real label, which reflects on the good score in the benchmark of the training set.

Now observing the test set, the result is significantly worse:

TEST SET - DIFFERENCES BETWEEN PREDICTED VS REAL



The average error on the test set is 16.69, almost 3 times the average error found on the training set. Again, it reinforces the benchmarking made on the test set.

V. Conclusion

Reflection

The summary of the project:
By using a dataset with personal and anonymous informations provided by the Brazilian company MeuCrediario, the data of 2500 hire purchases were used to train

a supervised learning model that predicts whether the customer will be a good payer or not.

In the first part of the project, I removed outliers regarding the label and also made clusters with a big categorical feature that the random forest model wouldn't have a good performance with.
Also in the data exploring part, I tried to reduce the dimensionality of the features by applying PCA, but although many number of eigenvectors were in fact selected, the end result got actually worse.

And in the second and final part of the project, I used a parameter searching tool to help tuning the final model and also tried many simple and ensemble regression models, but ended up with the best results possible using random forest regressor.

In the final result it became evident the presence of a high overfit in the model. My final conclusion is that even if applying dimensionality reduction the result did not get any better, we're dealing with variance in the data caused by the lack of training points. I was certainly bit by the curse of dimensionality.

But given the good score in the training set, the data seemingly make sense and this have potential to become a real world problem solver.

Improvement

As stated in the reflection session, the dataset must be bigger, way bigger for the model to be able to generalize the solution into unseen data. This is definitely the most important problem to be solved and without it this project cannot be used in production.

With more data, GridSearch will become eventually too slow to compute every time, so extracting the tuned parameters for further use it something to consider doing as soon as possible.