Machine Learning project

Wei WANG, Judith Jacquinot

January 16, 2022

1 Introduction

1.1 Project proposal

In this project, we are trying to use what we learned on the class, machine learning codes specifically, and other basic python codes to build a model. In this model, we would try to analyze our available dataset and predict the **target** values of another dataset with all the related information.

1.2 data background

In our project, we have one learn dataset and one test dataset and other relative files to make our personal imformation contain as much aspects as we can.Besides the current files we have,we add a last data set: PCS-ESE_2017_Liste.xls. It allows to make the link between the complete PCS-ESE codes (N3) and coarse profession groups (N2 and N1)

2 Preprocessing

2.1 Loading the data

In the data loading stage, we used pandas tool to load the csv format data, and we proceed to some reindexing of variables to prepare the merging stage.

2.2 Merging of the different datasets

We wanted to construct a single comprehensible and coherent dataset which would enable us to visualize and cross analyse easily all the variables available. To facilitate the merging process, we set as index the '**ID**' variable (enables to identify data relating to a single individual) and applied the join method for "additional information" type variables (sport). Concerning geographical data, we used the '**insee**' variable to identify correctly data relating to a single city. We then applied the merge method (using the'**insee _code**'column) in order to obtain a full dataset containing all the variables available, by merging the dataset city_adm, Region, city_pop, and Departement.

2.3 Handling missing values

- Several variables in our dataset were containing missing values:

'**sport**': describes to which sport club the individual belongs to
'**type_of _contract**': gives the working contract of the individual
ACTIVITY_SECTOR: economic sector of the job.
type_of_contract: work contract type.
JOB_CATEGORY: type of job (regular, internship, etc.)
Work_condition: job terms (full-time, part-time, etc.)
WORKING_HOURS: total annual working hours (this variable has missing values);

Employer_category: type of employers
Employee_count: size of the company
job_dep: department in which the job is located
Work_description: description of the job according to the PCS-ESE 2017 norm
Wage
EMP_CONTRACT : kind of job
Missing values are often tricky to deal with. Having missing values in a dataset can occur for different reasons:

1. Error in the data collection stage
2. Unavailability of the variable value for observations during the collection stage
3. Bad storage of the data

Concerning the variable 'type_of_contract', we observe by looking at the variable act, that the missing values corresponded to:

1. inactive people, or unemployed people. Hence, we decided to implement the value 'without contract actually' to these observations.
2. some active people, probably with another status than "employee". We decided to conserve the missing Values, and to treat this problem in the machine learning part.

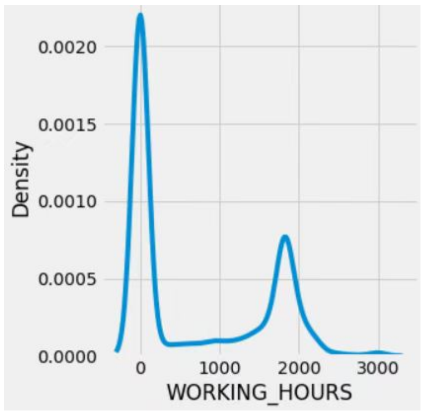
We observed that the missing values of the variables 'Job' variables came all from the same observations. We decided to:

1. Implement a value "without employ actually" to the categorical variables and attribute a value "0" to HOURS_WORKED and "wage", by deducing that inactive and unemployed people shouldn't receive a wage and work on the job market.
2. Keep a missing value for active people

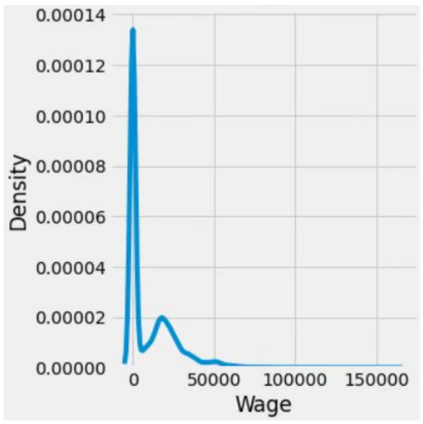
Concerning the variable "EMP_CONTRACT", we saw that no active and no unemployed people got missing value. Hence, we decided to implement a value 'inactive' to the missing value. Concerning the '**sport**' variable we figured out that the large number of missing values was probably resulting from the fact that many people were not part of any club. After that, we decided to impute the last missing values of the actives:

1. For the numerical values (Wage and WORKING_HOURS), we imputed the median values (since we had to delate some outliers after that) of the active (without considering the zero values that we imputed for inactives).
2. For the categorical values, we imputed the most common value of the actives, except for the variables type_of_contract. Indeed, we decided to impute for the independent worker a "entrepreneur" value. For the other, we kept the most common value through the active population.

Imputing the missing values with the median and with the most common value has some inconenients. Indeed, it underestimates the variance, and significantly distort the histograms. It should not change significantly the results since as we checked in the graphical part, data are not too much asymmetric.



(a) the distribution of working hours



(b) the distribution of wages

Figure 1: Results of loop function

2.4 A quick descriptive analysis

We can check our variables, and make several observations: The target variable is no skewed,

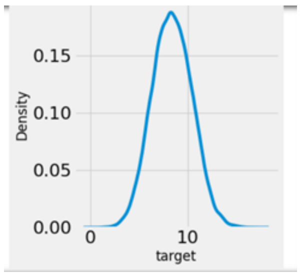


Figure 2: The distribution of target value

and we can see on the graph a nice normal distribution. Hence, our data set is not imbalanced.
We can see that the target variable is dependant of most of the other variables. We can

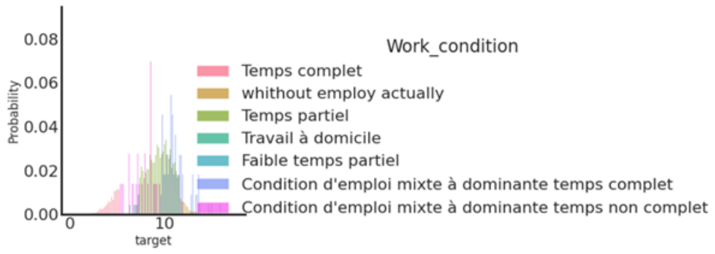


Figure 3: The distribution of target value with work condition

check it by looking at the variable city_type, JOB_CATEGORY, and Work_condition. These observations are made with a view to build a well stratified subsample.

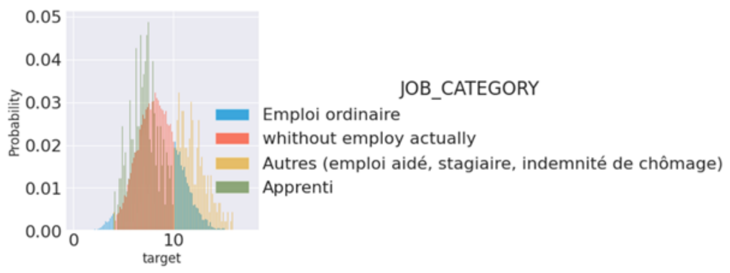


Figure 4: The distribution of target value with job category

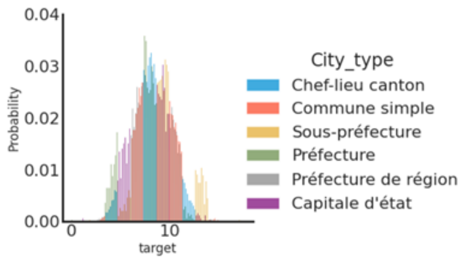


Figure 5: The distribution of target value with city type

2.5 Encoding of the variables

The reencoding of the variables was performed through three major parts:

1. Provide new nominations to categories
2. Recategorize some variables

2.5.1 Provide new nominations

The data obtained from the collection phase was messy, full of abbreviations and unclear nomina-tions. The first step of variable encoding consisted in providing some clear nominations to 3 each category of each categorical variable. This process facilitates interpretation during our graphical analyse of the data.

2.5.2 Recategorize some variables

- Encoding of sport variable:

As said previously, 'sport' variable contained a lot of missing values. We supposed that people with missing observations simply didn't below to any sport club. So, we decided to simply create a variable 'Club'. If people belong to the club, we assign the value "TRUE". Otherwise, we assigned "False".

- Encoding of the arrondissements, and their populations:

We observed that it was not possible to study well the size of the big metropoles (Paris, Lyon, and Marseille), because the zip code (variable 'insee') was linked to the arrondissement, and not to the cities themself. Hence, we decided to reencode the arrondissement, and the INHABITANTS value of those arrondissement. In order to reencode, we used the method **.replace()**,on the variable 'Nom de la commune', to replace the name of the arrondissement by the name of the city. To replace the population of the arrondissement by the population of the city we filtered our dataset using the variable 'Nom de la commune' and the method **.startswith()**, to replace 'INHABITANTS' by the sum of the population of all the arrondissements of the city.

- Encoding of the variable job_dep:

After having a look on the distributions of the variables, with saw that job_dep was totally unreadable. We know that there is probably a strong collinearity with 'dep', du to the fact that most of the people work in their own department of leaving. Hence, using the methode np.where, we choose to create a new variable 'job_localisation_departement', with tree values:

1. 'same departement', if the individual leave and work in the same department
2. 'other departement', if the individual leave and work in different department
3. 'inactive', if the individual is identified as inactive (unemployed or a real inactive)

- Encoding of the variable variable Work_description

The variable Work description is precise, and it will probably be a problem, since a lot of values correspond to few people. We decided to simplify that, by creating another csp variable, corresponding the less deep level. We used our imported data set 'csp_mapping.csv', and by using the first letter of the string in the Work_description value, and then by using the method .replace(), we found a way to represent the less deep level of the csp.

2.6 Dealing with the outliers

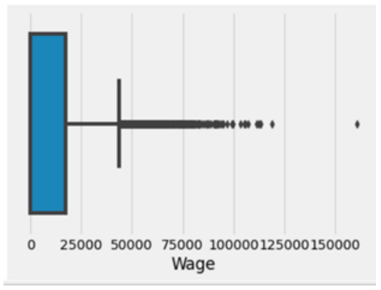


Figure 6: The boxplot of the wage variable

After having a check on the boxplot of the numerical variables (Wage, age_2020 and WORKING_HOURS), we decided to try to delate the outliers. Delating the outliers can be a good thing, since CORRELATION COEFFICIENT is highly sensitive to outliers. In order to delate the outliers, we used an univariate analysis, and we used some boxplot to represent them. Then, we delated the values out of the theoretical low and up limits. However, by checking our results in the machine learning part, we saw that the scores of the several models were lower, when delating the outliers. It's maybe du to the important loss of information. Hence, we didn't run this part of the code, but you can still check it (we just transformed it in a text part).

2.7 Delating some variables: A first selection feature

At the end of the day, we chose to delate some variables. Indeed, a variable with : Sports, which contain a lot of missing values. All the information are contained in 'club' variables

1. 'insee': The INSEE information's are already contained in the variable "nom de la commune"
2. "Reg", which gave the same information than "nom de la region"
3. "Dep", which gave the same information than "nom du département"
4. Job_dep: as we already said, there was a strong collinearity with the variable 'dep'. This collinearity could reduce the predictive power of the linear model.

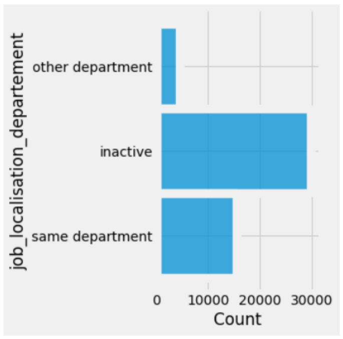


Figure 7: The distribution condition of new created variable

By the way, later in the code, we checked the variance of our variables, in order to delate variables with a too low variance (this low variance could induce outperformance in our machine learning algorithm). We use the module Variance Threshold, but after checking, no variable had to be delated.

2.8 Using specific encoder

In order to get useable values in our different kind of models, we need to encode them, with specific tolls. Using well-chosen encoder tools has several consequences:

- We can use categorical variables, which are string or objects, by converting them in integers
- We can make our algorithm more performant, by normalizing our numerical variables

In order to normalize the numerical variables, we use specifically RobustScaler. Indeed, we got some variables with a lot of extremes values, as HOURS_WORKED, for extend. Using Standard Scalers would induce our machine learning algorithms to mix up some values. In order to encode our categorical variables, we would prefer to have using OneHotEncoder, which has the advantage to not create a fake range our categorical variables. However, du to difficulties of coding, we have chosen to use LabelEncoder. Label Encoder will probably create some bias in our linear models, but not in our decision tree and in our random forest models. However, it has an advantage: we can use some tool of feature selection with labelEncoder, and no with OneHotEncoder, specifically as RFECV or RFE.

3 Machine Learning Algorithm

3.1 Preliminaries

3.1.1 Set apart the target variable from the other variable

We used the methode .drop, to create our predictive features data set: X and a target variable vector: y.

3.1.2 building a stratified subsample

Building a subsample allows to run the code much faster, and to debug it. However, we need to get a well stratified subsample. The main advantage of stratified random sampling is that it captures key population characteristics in the sample. Hence, it allows to get almost the same results than the main dataset, when running the machine learning algorithm. After looking at the dependence of the variables, as previously said, we noticed that almost all the target variable was dependant of all the variables. Hence, after failing of building a sample set stratified on all the other variables, we decided to build this sample based on the stratification of the target variables.

3.1.3 Simple selection feature based on the variance

We computed the variance of the variables. Delating variables with too low variance allow to get better predictions.The idea was to removes all features which variance doesn't meet some threshold. We choose a threshold equal to 0.3. Finally, we didn't have to remove any feature.

3.2 Linear Models

3.2.1 Preliminary: selecting a good sample size

By minimizing the Mean Squared Error, and miximizing what should be the score of a Linear regression, we decided to take a proportion of 1/3 of the all sample, for the training set. Taking a too important part of the data set could induce overfitting, but a too small train set could reduce our chances of getting good results. Indeed, our algorithm could miss information. And we build our train and set samples thanks to the method train_test_split.

3.2.2 Linear regression

We first decided to use a first linear regression.

- By using a Kfold, we found an internal cross validation mean squared error of 1.8771559276614027 and a R-squared of 0.5527992053515713
- By comparing our prediction, and our test set, we found a first Mean Squared error of 1.8928077430587602, and a R-squared of : 0.5450531641909667

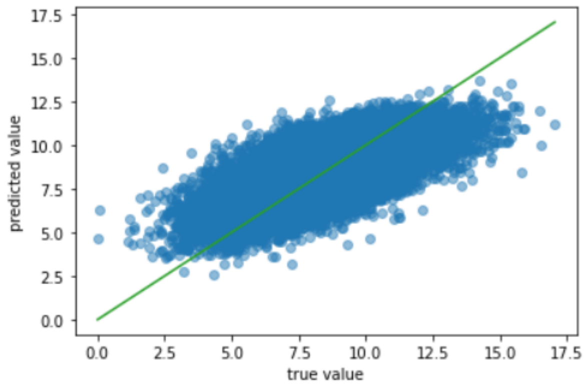


Figure 8: The linear regression curve

We decided to use a grid search, with a feature selection, in order to find the right features. We use as estimator, RFE (from the module) , applied to the linear regression. The parameters we selected was then number of features. The principle of the RFE function is to remove the less useful feature, each time it run, and allows to select the feature, to maximize the score of the model. What we fin d is that, after having taking 20 feature, we don't really improve our score (here, a r-squared). Finally, by comparing our predictions and our test

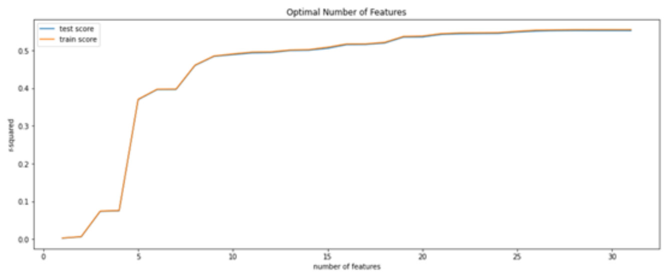


Figure 9: The R-squared score of the linear model

set, with 20 features, we found a r-squared of 0.5450531641909667, and a mean squared error of 1.8928077430587602, which is a little bit less good than with all the feature, but not so much.

3.2.3 Ridge Regression

We try to use a ridge regression, then, to optimize our model.The ridge function can be useful, when variables are correlated. Indeed, it allows to put a quadratic weight, to some coefficients, function of the correlation of the variables. Here, we got thanks to a first ridge regression, with a default parameter alpha=1,

- An internal MSE od -1.8764861500515475, and a R-squared of 0.5527924530320458
- a MSE equal to 1.89243393453213, and a R_squared of 0.545143011142942, by comparing our prediction and our test set.

Those results are a little bit better than the results of the linear regression, with all the feature. By looking at a validation curve, we see that our score is better when alpha is low. Then, we tried to select the best parameters, for the function ridge, with a grid search. We finally found with the several parameters:

- 'alpha': 0.0001
- 'fit_intercept': True
- 'normalize': False

And we get the result of the best score for 0.5529607333698039,and r2 score: 0.5451511564860487. Then, we tried to use a polynomial feature. The idea was to add polynomial feature of degree 2, for the numerical variables. At first we wanted to use a pipeline, to compute all the best parameters, and the best degree of feature, but it needed to much time. Rather, we decided to use the method fit_transform, with as transformer PolynomialFeature(degree=2). By doing a grid search, and using a Kfold as cross validation, we found as best parameters:

- 'alpha': 0
- 'fit_intercept': False
- 'normalize': True

And we calculated our best score:0.57830893342427292 , and by using this model on our test set, we found r2 score: 0.576964131014916. Finally, we decided to use a pipeline, and to look for the best parameters and feature, by applying the estimator RFE to the ridge regression, thought a pipeline. We used the data set with the polnomial features. It gave us different results:

- It proposed us as best parameter an alpha of 100
- Gave us an internal cross validation score of 0.544, based on R2
- A score on the test set of 0.541, based on R2

Ideally, we would have use another "clean" data set, without applying LabelEncoder, without Robust Scaler, without imputing missing values, and we would have built a pipeline, with:

- A preprocessor composed of OnehotEncoder, and RobustScaler, built with ColumnTransformer
- An IterativeImputer, to impute some values with the missing values
- A PolynomialFeature
- The model with wanted to use (here, Ridge)

This proposition would have probably been better to select the best model, and would have im-prove our preprocessing step. However, du to difficulties, we were not able to propose this solution.

3.2.4 Lasso regression

The Lasso regression allows by itself to simulate selection feature. Indeed, the parameter alpha controls the strength of the penalty.When alpha is equal to zero, no parameters are eliminated, and when alpha tends to infini, all parameters are eliminated. However, as alpha tends to zero, as the biais increases.Here, we will test our model with a Lasso regression, and see if it improves our predictions.We first test our Lasso regression with the parameter alpha=1.

- We got a bad internal cross validation mean squared error: 4.130997835704742, and a bad r-squared of 0.007091765669360206
- By comparing the prediction and the test set, we got an MSE of 4.136576886244636 and a r-squared of 0.007531941347091786

Those results are bad, and maybe due to the parameters: we decided to use a grid search. By doing a grid search, we found as best parameters:

- 'alpha': 1e-05
- 'fit_intercept': True
- 'normalize': False
- 'positive': False
- 'selection': 'cyclic'

An internal cross validation r2 score of 0.5528088582994862,A R2 score on the test set of 0.5460470768760796. We also tested, as for the Ridge regression, a polynomial feature. The best model seems to use the following parameters:

- 'alpha': 0.0001
- 'fit_intercept': True
- 'normalize': False

- 'positive': False
- 'selection': 'cyclic'

So we got our best score with the value of 0.5528777596638071. We got also some other scores, on the test set:

- 1.8922006605533188 for the mean squared error
- 0.5451990798371482 for the R-squared

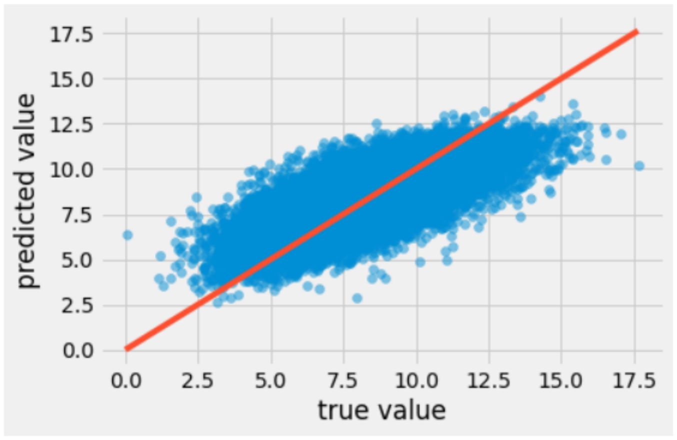


Figure 10: The regression curve of lasso model

3.2.5 Conclusion of the linear models

It appears that the best model is the ridge regression, without normalization, an alpha equal to zero (which means that we don't have to put any quadratic weight on the parameters), and a fit interception.By the way, it appears that using polynomial features of degree 2 increases substantially our score, more than using another model, and other parameter.However, we could probably improve our result with a well-done pipeline, and notably a better preprocessing.Finally, we can ask the following question: how much data do we need to improve the following model?We did in order to answer to this question a learning curve. What we can notice is that our internal score doesn't increase with the train size. We could also say that we don't really need so much data, since the score of the test size stop to improve after 8000 elements.

3.3 The K-nearest neighbours

3.3.1 A simple knn model

We build a simple K-nn model, by looking at the best nearest neighbours.We can first say that, by looking at the mean squared error, the internal cross validation score is bad with a few number of neighbours, and begins to be stable, with 10 neighbours. By minimizing the Mean

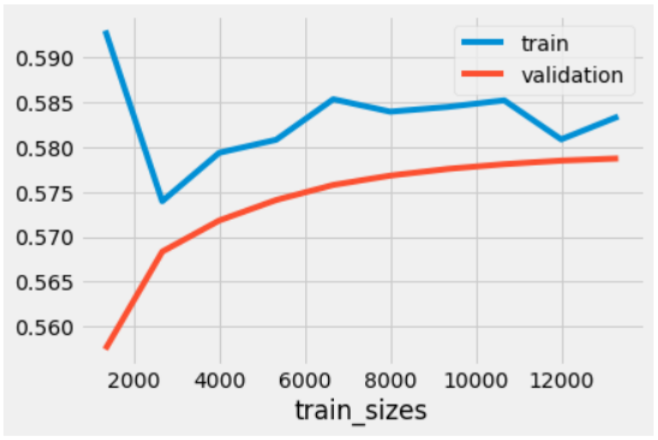


Figure 11: The learning curve of ridge regression with polynomial feature

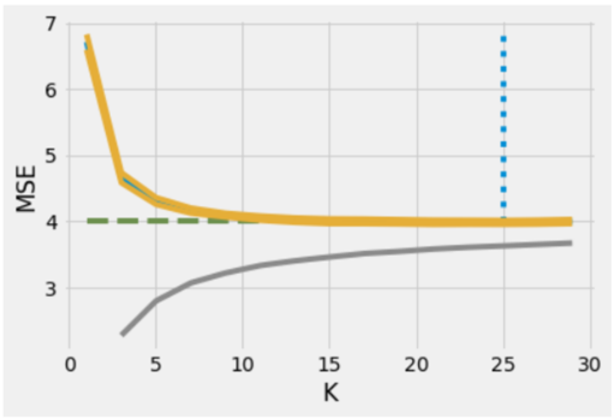


Figure 12: The relationship between MSE and K value

squared error, it seems that, the best knn is 25, as seen on this graph, with an internal mse of 3.9837538417669784. By choosing this neighbour, we get a MSE of 3.9490972805113405, and a R-squared of 0.05251297891897089. We can see that these predictions don't fit well the data of the test set.

3.3.2 Polynomial features

We use the same data set that we build, in order to look at the add of the polynomial features. We got as best knn : 19 with a mean squared error 3.94072452475031164, by looking at the internal cross valuation score. By testing it predictive power on the test set, we got a MSE of 3.94936052162795, and a R-squared of 0.05553947843157092.

3.3.3 Conclusion of the Knn method

It seems that the KNN are less appropriate in our case. That's not surprising, since the knn are not good in order to compute the best model when there is a big sample set.

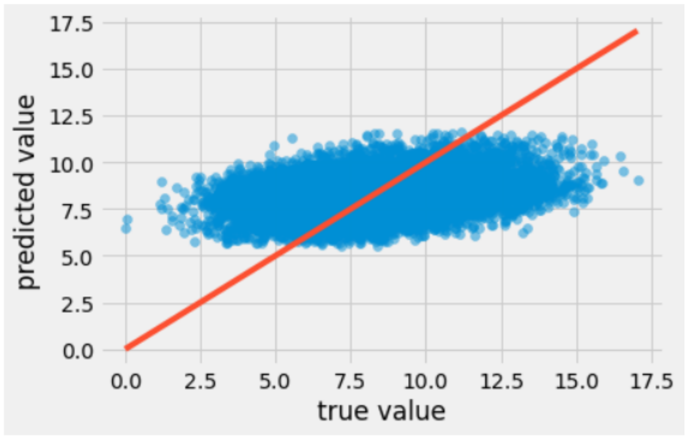


Figure 13: The relationship between predict value and true value

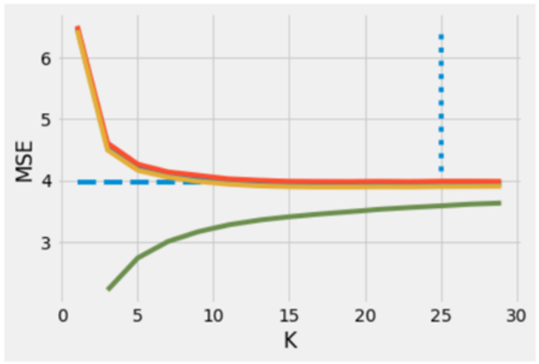


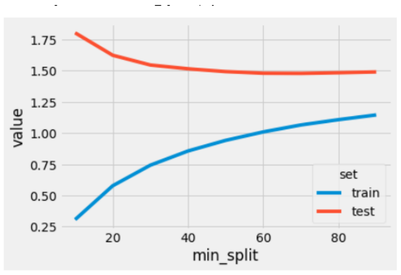
Figure 14: The relationship between MSE and K value under polynomial features

3.4 Decision Tree model

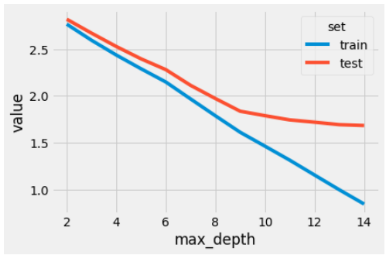
There is another ay we would apply here is decision tree to predict our target results. As what we did in linear regression model for dealing with data. In decision tree and later random forest model, we would adjust our dataset to satisfied our model. The first thing we did is to delete those variables pointless with very obvious reason that we don't even need to use, like those geographic information. And then we translate those text information into dummy variables, like 'club','wage' and so on. The first model we are trying to do is to use the decision tree model. If we don't do any optimization, we would get a very sub-optimal model. So we need to choose better parameters in our model. At our first attempts, we use a loop function to select them. And as we did and show, it's not a very efficient wat to manage it. So we turned to GridSearch CV methods to find our best parameters.But there is one thing that maybe could help us get rid of this long process.That is the random forest model.

3.5 Random forest model

Different from decision tree model, random forest model is another very useful model to predict the results. Especially that it could run many decision trees at the same time. So it supposed to have better performance theoretically. But we will see it after our performance.



(a) performance for min split



(b) performance for max depth

Figure 15: Results of loop function

So apparently we should choose random forest model to better help us predict the results.

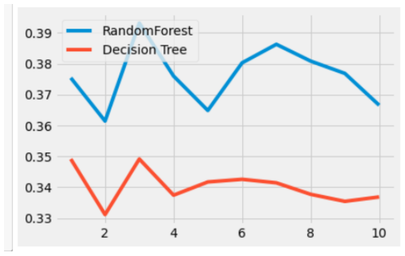


Figure 16: The performance of decision tree and random forest

There is another issue to be determined that if we should use 'RandomForestRegressor' or 'RandomForestClassifier'. The difference between these two models is not so big. But luckily, we could still use the similar way as we just did, using cross val value to distinguish them. We could see that 'RandomForestRegressor' is strongly better. So based on it, we would use

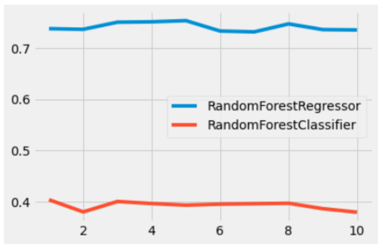
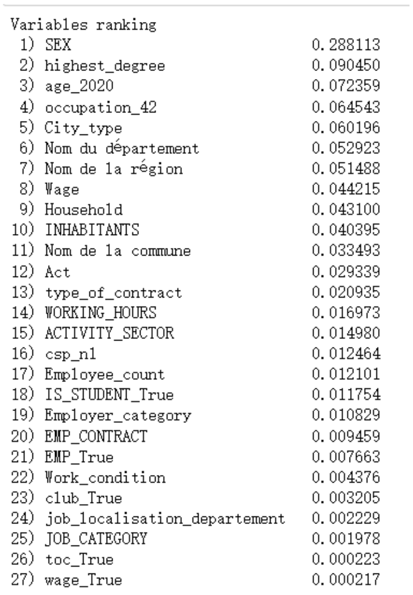


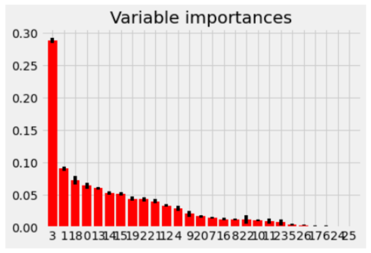
Figure 17: The performance of different random forest model

'RandomForestRegressor' in the coming project. Once we decide our model, we could start to optimize our model.Similarly,we do the same thing as we did in the decision tree part. We build a very basic model with nothing restricted. But now we won't use loop functions to choose our best parameters, we would use GridSearch CV methods to optimize our model directly. But before we do it, we can first check the importance of features in our original model to see if there are some variables that are totally useless. We can see from the figure that there are several variables barely contribute to our target value. So we just delete them since then to simplify our model, reduce the calculation time and precise the results. Based



```
Variables ranking
 1) SEX                              0.288113
 2) highest_degree                   0.090450
 3) age_2020                         0.072359
 4) occupation_42                    0.064543
 5) City_type                        0.060196
 6) Nom du département               0.052923
 7) Nom de la région                 0.051488
 8) Wage                             0.044215
 9) Household                        0.043100
10) INHABITANTS                      0.040395
11) Nom de la commune                0.033493
12) Act                              0.029339
13) type_of_contract                 0.020935
14) WORKING_HOURS                    0.016973
15) ACTIVITY_SECTOR                  0.014980
16) csp_n1                           0.012464
17) Employee_count                   0.012101
18) IS_STUDENT_True                  0.011754
19) Employer_category                0.010829
20) EMP_CONTRACT                     0.009459
21) EMP_True                         0.007663
22) Work_condition                   0.004376
23) club_True                        0.003205
24) job_localisation_departement     0.002229
25) JOB_CATEGORY                     0.001978
26) toc_True                         0.000223
27) wage_True                        0.000217
```

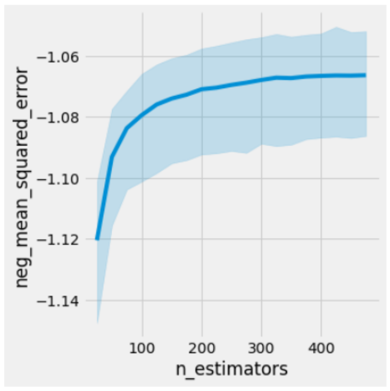
(a)



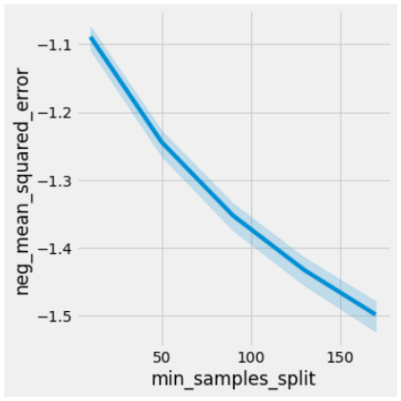
(b)

Figure 18: The importance of variables

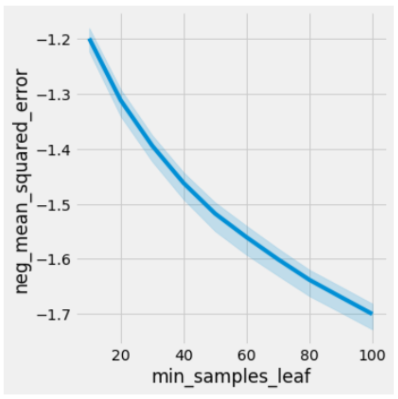
on the features of the model, we have a lot of choices to optimize our model.And we choose five features from them which would influence the model most, instead of testing them one by one to waste out time. As we repeat the function for several times for different features. We finally get our best model of 'RandomforestRegressor'. And we could see that during the process, the performance of mean squared error and r2 score performed better and better. We can see from above that our model is getting better by each time we applied the new best parameter.Such optimization is preformed by the changement of MSE and r2



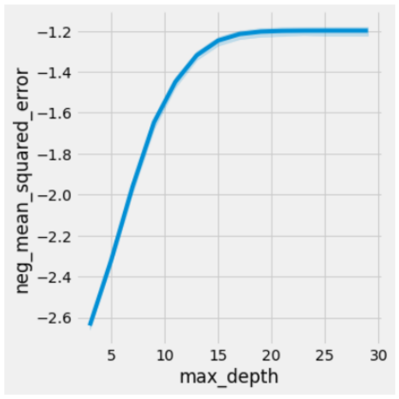
(a) performance for min_samples_split



(b) performance for min_samples_leaf



(c) performance for max_depth



(d) performance for max_depth

Figure 19: The optimizing process

4 Conclusion and prediction

In this report, we tried many methods to build and optimize a model to predict our target values. The standards we used to identify if it's a good model to predict or not are the coefficient of determination (r-squared) and of the mean squared error. So that based on the results we get above,we finally choose 'RandomForestRegressor' model to predict with best parameters we can use by selecting from the GridSearch CV. The prediction file is made to show off our results.