FTEC4003 Data Mining of FinTech

Course Project Report

Group 6

KEI Yat-long

So Cheuk Yin

Wong Chun Yu

Brief description of the platform:

As a high-level, general-purpose programming language, python provides us a wide range of packages to develop data mining models and visualize data. Therefore, we used Python 3 as main platform to perform data mining tasks in this course project.

Package imported:

- sklearn: responsible for implementing different data mining methods, as well as performing data preprocessing.

- pandas: responsible for importing and exporting data from / to csv, as well as processing data

- matplotlib: responsible for graph plotting for parameter optimization.

- xgboost: responsible for the XGBoost Classifier in Task 2.

TASK 1: Insurance Selling

The data comes from clients of an insurance company. These clients have already bought the medical insurance. Nowadays, the company wants to launch new transportation insurance and to find those who will be interested in this insurance. The data is related to an insurance selling problem. The clients' information is about clients' basic information and their vehicle's situations.

The classification goal is to predict if the client will buy the transportation insurance (i.e, identify the value of feature 'Response', 1 for yes and 0 otherwise).

Data Preprocessing

As the features "Gender", "Vehicle_Age" and "Vehicle_Damage" are non-numerical, we map them into numbers:

Feature	Orginal Value	Assigned Value
Gender	Male	0
	Female	1
Vehicle_Age	< 1 Year	0
	1-2 Year	1
	> 2 Years	2
Vehicle_Damage	Yes	1
	No	0

Methods

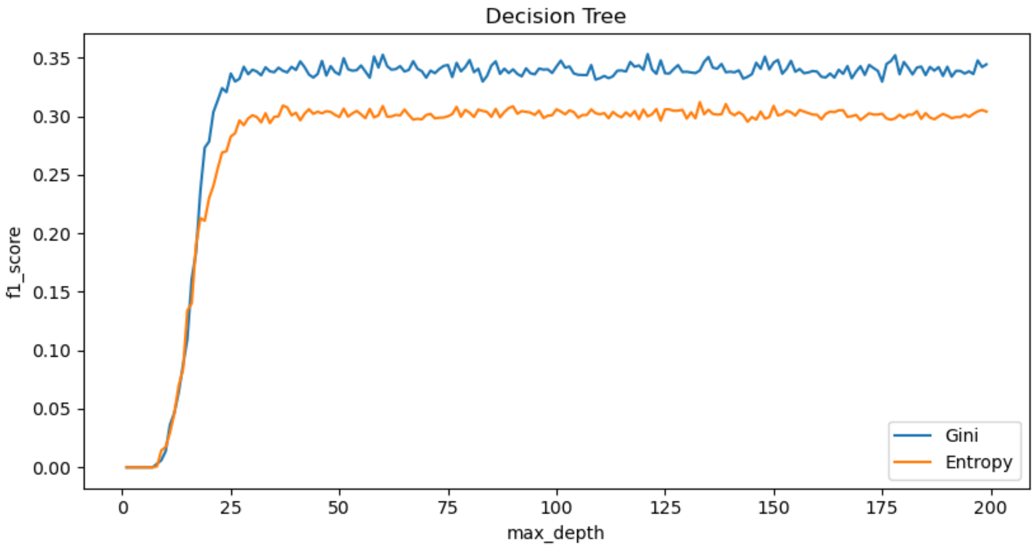
Below shows how we use all the 5 methods learned in class (Decision Tree, KNN, Bayes, SVM and Ensemble Method) to predict their performances.

DecisionTree

Experimental evaluations:

For features, we select all of them in our model since the decision tree model will select suitable features based on impurity score.

Decision tree with gini and entropy criterion for different max_depth:



When we use the gini index, model performance is better for max_depth >= 20. In particular, when using the gini index with max_depth = 82, 89, 174, 189, f1-score can reach 0.35.

Results:

We select all features, choose the gini index as criteria of the decision tree model and choose max_depth = 121, f1-score of **0.35309** can be achieved.

k-Nearest Neighbor

Experimental evaluations:

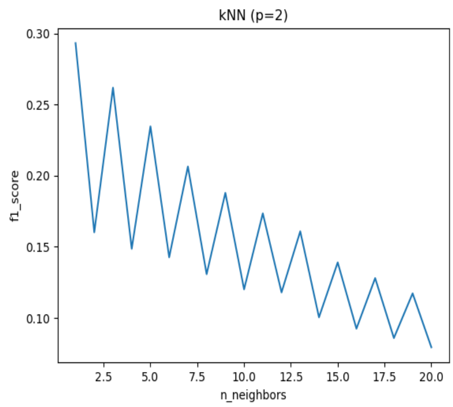
For data preprocessing, we first used PCA to reduce the data’ s dimension. And then, we applied preprocessing.scale to scale down the datasets so that kNN can run smoothly and perform better.

By human testing, when n_component = 8, the model performed the best. One interesting observation is that, after applying PCA, the model performs better for a small number of neighbors.

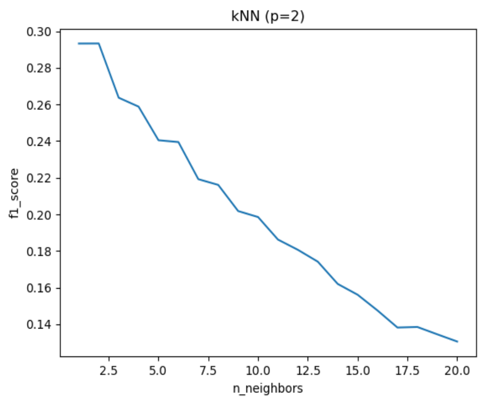
For feature selection, we remove irrelevant feature “id” as it does not contribute much for f1_score.

The two graph below shows how kNN model performs with 2 different methods to evaluate the distance of the testing records.

Method 1: weights= “uniform”



Method 2: weights= “distance”



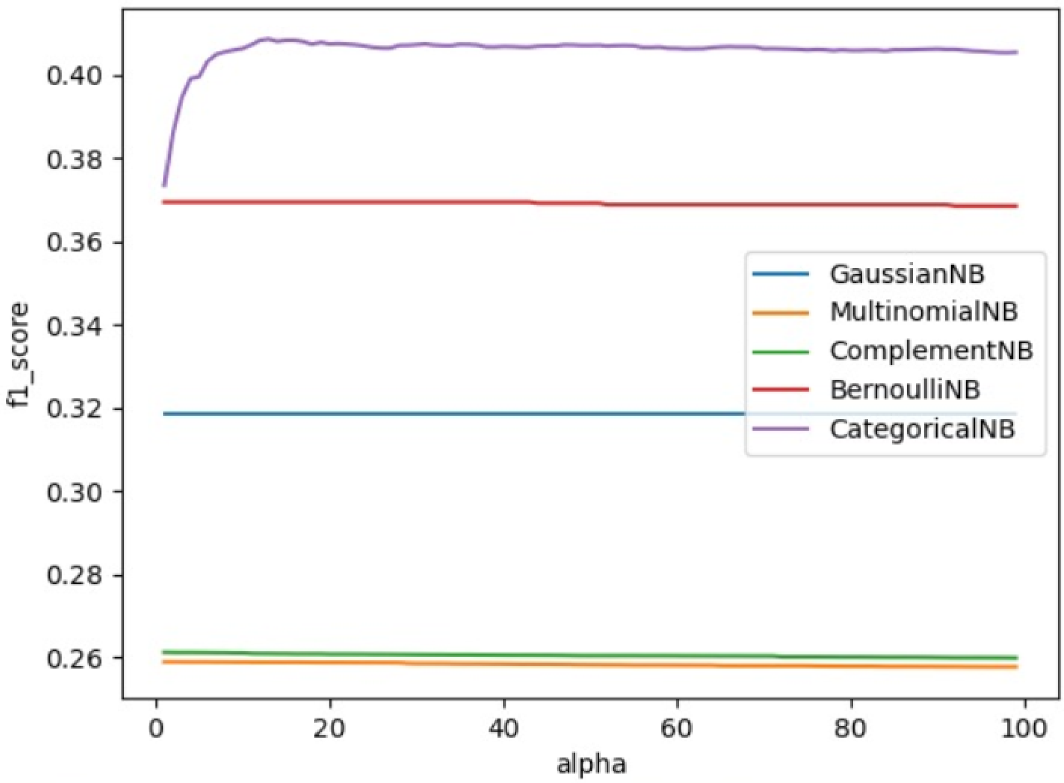
Results:

We use PCA and preprocessing.scale for data preprocessing methods. Select all features without “id” and choose n_component = 8, number of neighbors = 2, weights = “distance” , then we can get 0.2933047670058918 on f1_score.

Bayes

Experimental evaluations:

In sklearn, there are 5 different Bayes classifiers. When alpha is a positive number, the Gaussian NB is the only Bayes classifier without Laplace while the others are 4 different Bayes classifiers with Laplace.



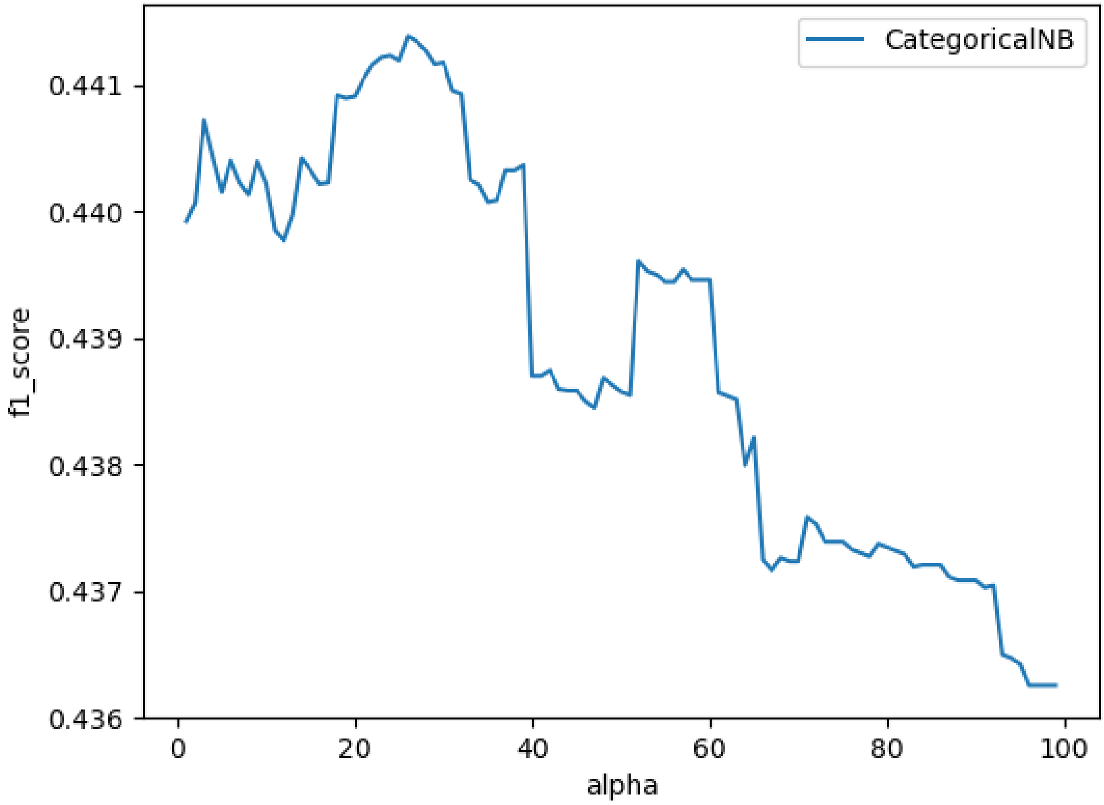
(Note: GaussianNB does not have parameter alpha.)

Before doing any feature selection (using all given attributes without "id"), we first investigate into which NB classifier performs the best.

From the graph above, it is obvious to show that CategoricalNB performs the best with different choices of alpha. Therefore, we choose CategoricalNB (with Laplace).

For feature selection, we only choose attributes "Age", "Previously_Insured", "Vehicle_Damage" and "Policy_Sales_Channel" by human testing..

Then, we tune alpha from the range 1 to 100 to see how it will affect the f1_score of Categorical NB (see the graph below):



From the graph above, when alpha = 26, CategoricalNB performs the best.

Results:

When we use CategoricalNB (with Laplace) , select 4 features "Age" , "Previously_Insured" , "Vehicle_Damage" and "Policy_Sales_Channel" , and choose alpha = 26, the corresponding f1_score is 0.44138785625774474.

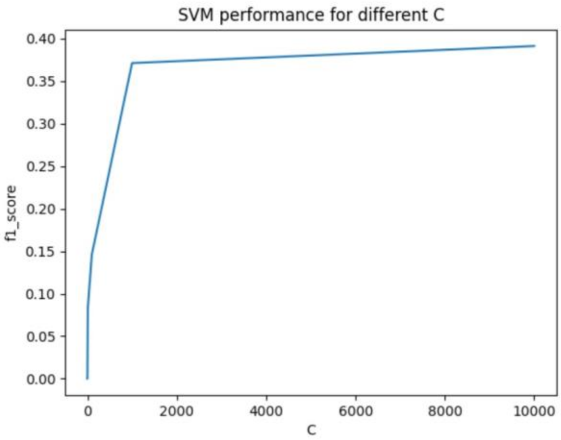
SVM

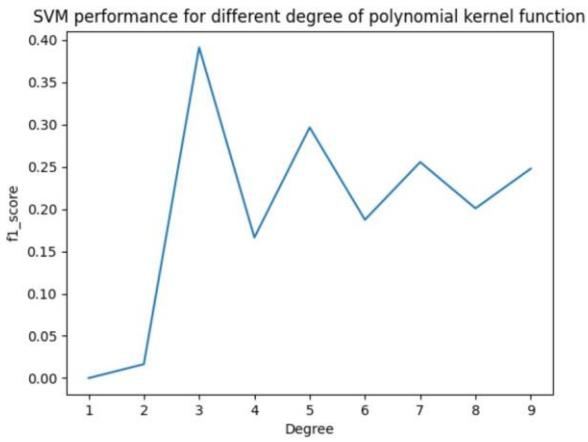
Experimental evaluations:

Size of training record: 540 (so that the training process can be finished in a shorter time and is suitable for parameter testing.)
Data preprocessing method: preprocessing.scale()

For feature selection, we only choose the attributes "Age", "Region_Code", "Previously_Insured", "Vehicle_Age", "Vehicle_Damage", "Annual_Premium" and "Vintage" by human testing.

Parameter:
C (Regularization parameter), Degree (of polynomial kernel function)





We can observe that f1-score keeps increasing when C increases.
The result starts to converge after C exceeds 1,000.

Since it may need excessive running time if we use a very large C, therefore we choose C = 10,000 to make a balance.
And we can get the highest f1-score with Degree = 3.

Results:

We can get the highest f1-score of **0.398** with reasonable runtime at C = 10,000 and Degree = 3.

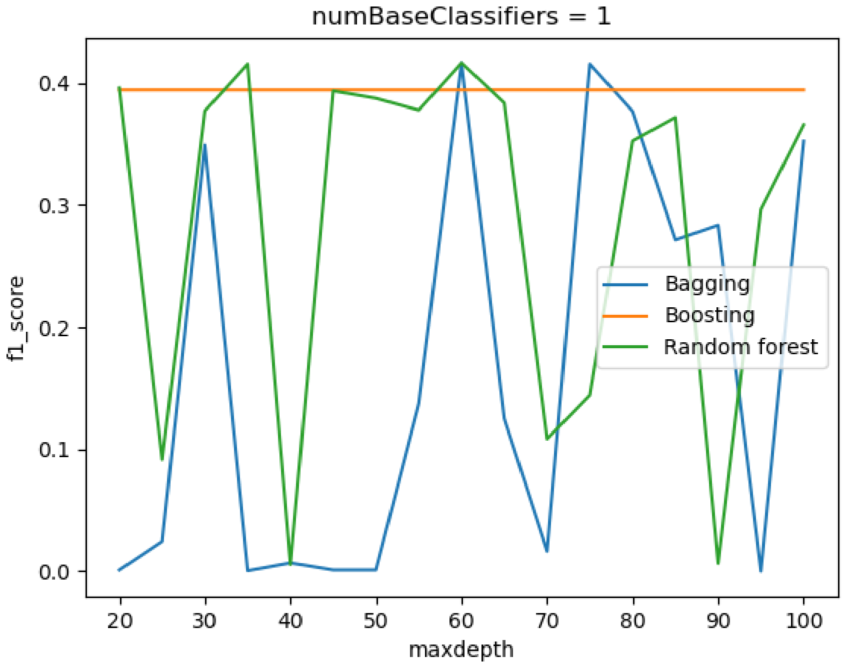
Ensemble Method

Experimental evaluations:

For data preprocessing methods, when we use preprocessing.minmax_scale on the training data but do nothing with testing data, f1_score raises from 0.3 to 0.4. Therefore, we will use sklearn.preprocessing.minmax_scale to scale down the range of the training datasets only.

By human testing, we remove features "id" in our testing since it will affect the performance of the ensemble models.

First, let us find out which ensemble method performs the best with given numBase Classifiers and different choices of maxdepth.See the graph below:
Testing parameter: maxdepth

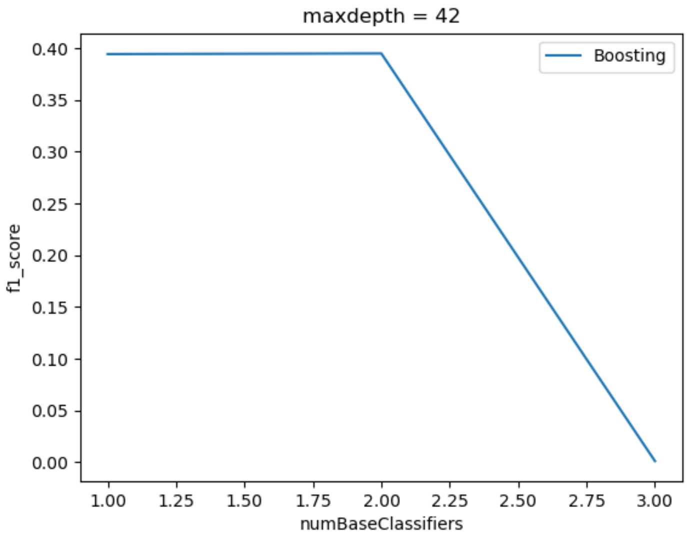


(Note: Random forest does not have parameter maxdepth.)

Although both Bagging and Random forest can obtain over 0.4 f1_score at same maxdepth, we select Boosting to investigate since it is risky for a random state.

Let’ s see what number of numBaseClassifiers will make the ensemble methods perform the best with given maxdepth = 42.

Testing parameter: numBaseClassifier



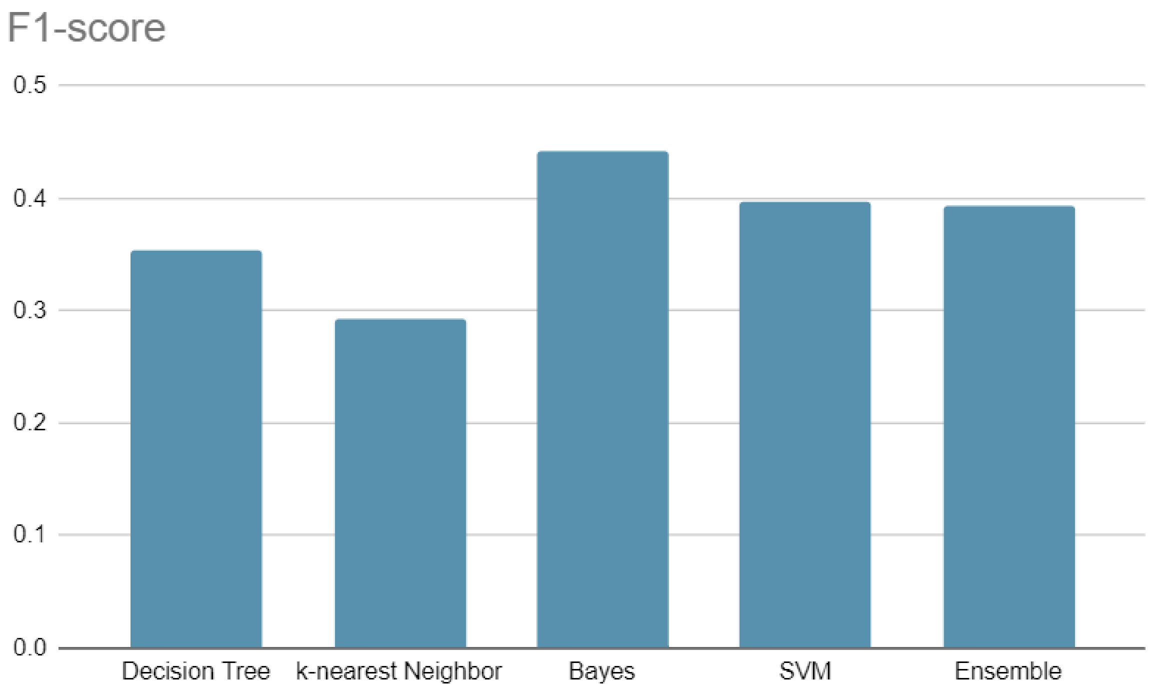
We can get the highest f1-score with numBaseClassifier = 1.

Results:

We can get the highest f1-score of around **0.3943938767066611** at maxdepth = 42, numBaseClassifier = 1 with using Boosting.

Conclusion:

Methods	f1_score
Decision Tree	0.35309
k-nearest Neighbor	0.2933047670058918
Bayes	0.4413878562577447
SVM	0.397724039829303
Ensemble	0.3943938767066611



With the optimal parameters choices, both Decision Tree, k-nearest Neighbor and Ensemble methods can obtain f1 scores ranging from 0.29 to 0.394, while the best 2 methods are Bayes (with Laplace) and SVM since they can even obtain f1-score higher than 0.397.

TASK 2: How many Customers Stay

The data comes from clients of a bank. These clients have already had accounts in this bank. Nowadays, the bank wants to model whether they will stay or not in the future. The task is to do the binary classification based on the given information, which gives extra information to the bank to stabilize the customers. The data are attributes of customers' basic information.

The classification goal is to predict if the customer will leave this bank and choose other competitors in the future (i.e, Identify the value of feature 'Exited', 1 for yes and 0 otherwise).

Data Preprocessing

As the features "Geography" and "Gender" are non-numerical, we map them into numbers:

Feature	Orginal Value	Assigned Value
Geography	France	0
	Spain	1
	Germany	2
Gender	Male	0
	Female	1

For "Surname", unfortunately, we cannot find a good way to map them that performs well[1]. Besides, "surname" in nature is kind of irrelevant to whether the customer will leave the bank. Therefore, we would exclude them from the data.

[1] We indeed tried many ways to map data of "Surname" (e.g. map according to alphabet), but they don't seem to pe rform well.

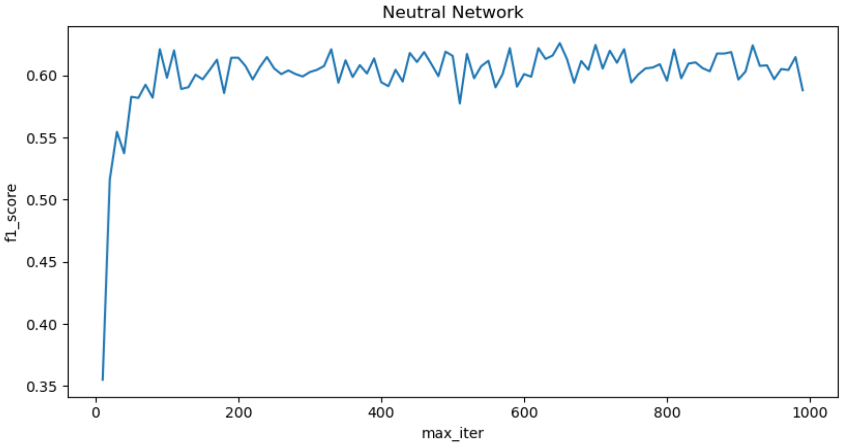
Methods

Below shows how we use most of the methods covered at Task 1 (Decision Tree, Bayes, SVM and Ensemble Method), as well as some other methods we learnt in lectures and tutorials to predict whether the customer will stay or not in the future.

Artificial Neural Network

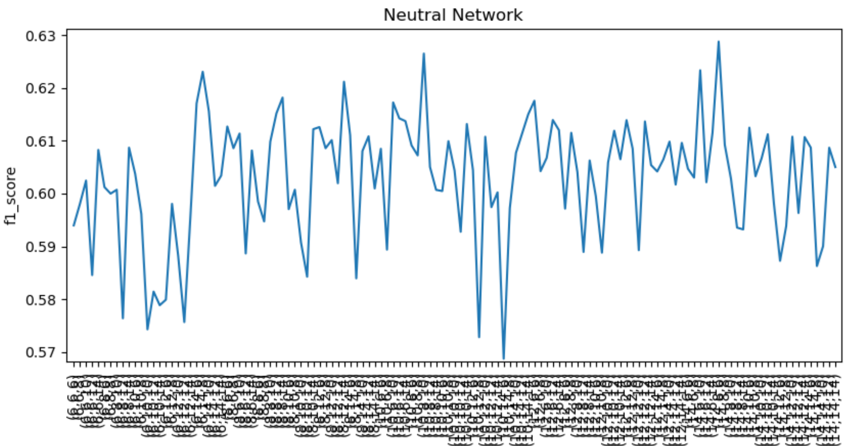
Experimental evaluations:

hidden_layer_sizes=(10,10,10),solver='adam', activation='tanh'



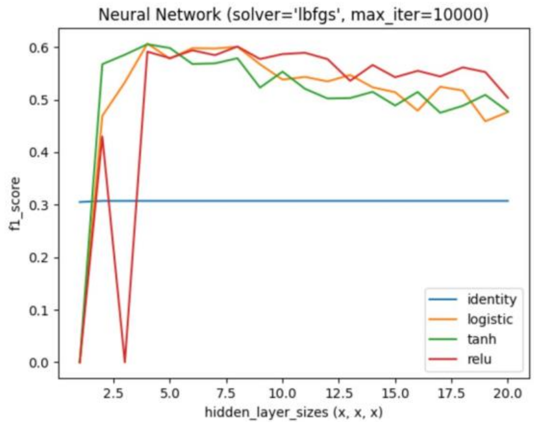
param_x, param_y, param_z = [6, 8, 10, 12, 14]

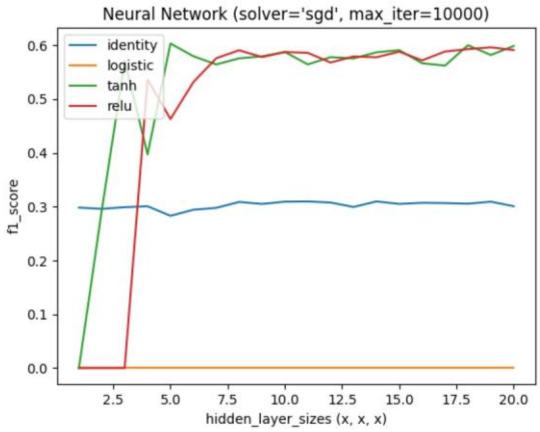
hidden_layer_sizes=(param_x,param_y,param_z),solver='adam', activation='tanh', max_iter=500

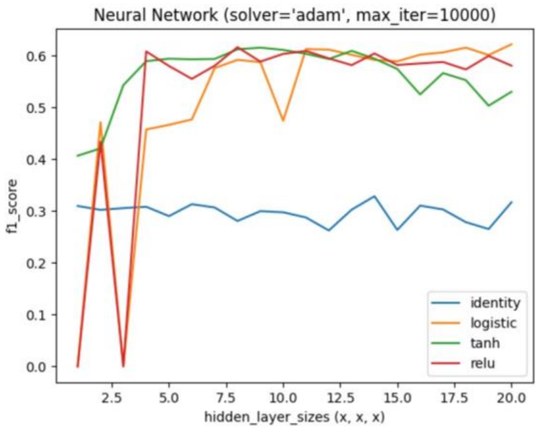


It seems that the performance is similar for slightly different hidden_layer_sizes.

Testing on different solver and activation function:







Results:

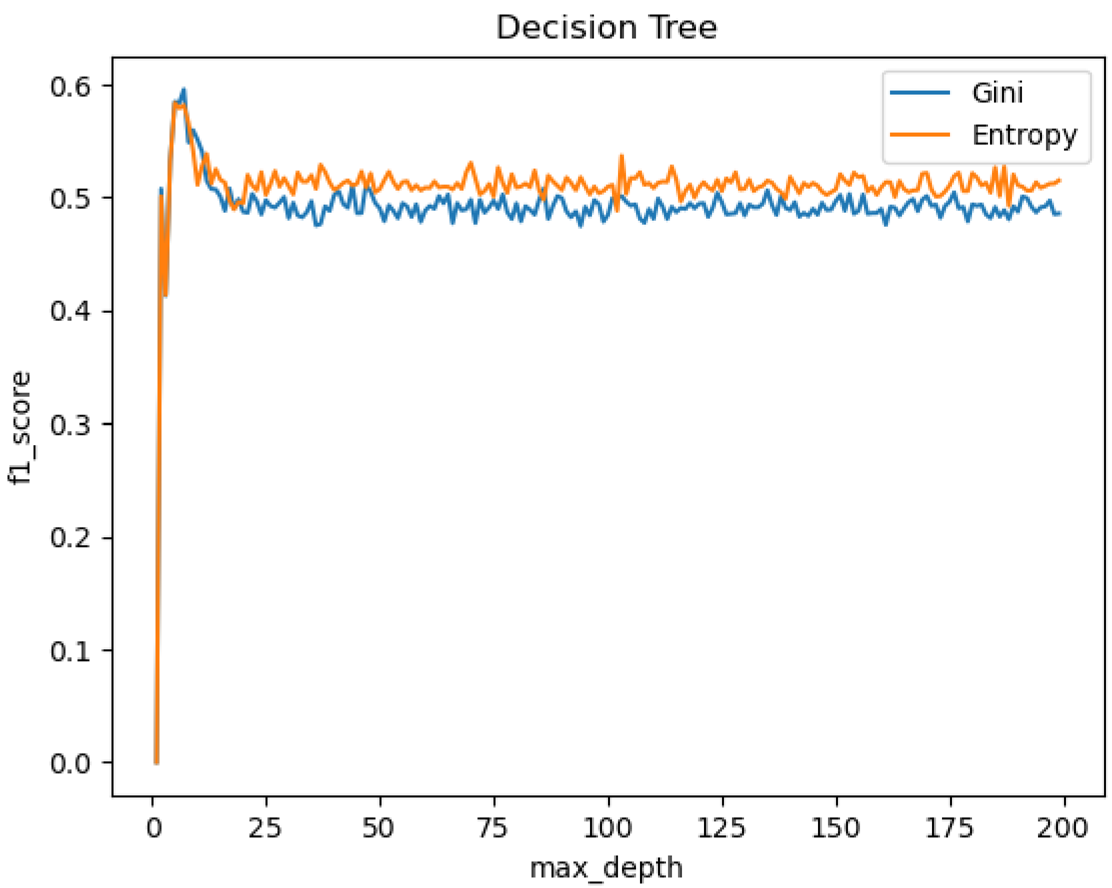
When we use preprocessing.scale to preprocess the data, select all features except "CustomerID", "Surname" and "RowNumber". And then, we set hidden_layer_sizes=(20,20,20), solver='adam', activation='logistic', max_iter=10000, the f1-score can be **0.6216**.

Decision Tree

Experimental evaluations:

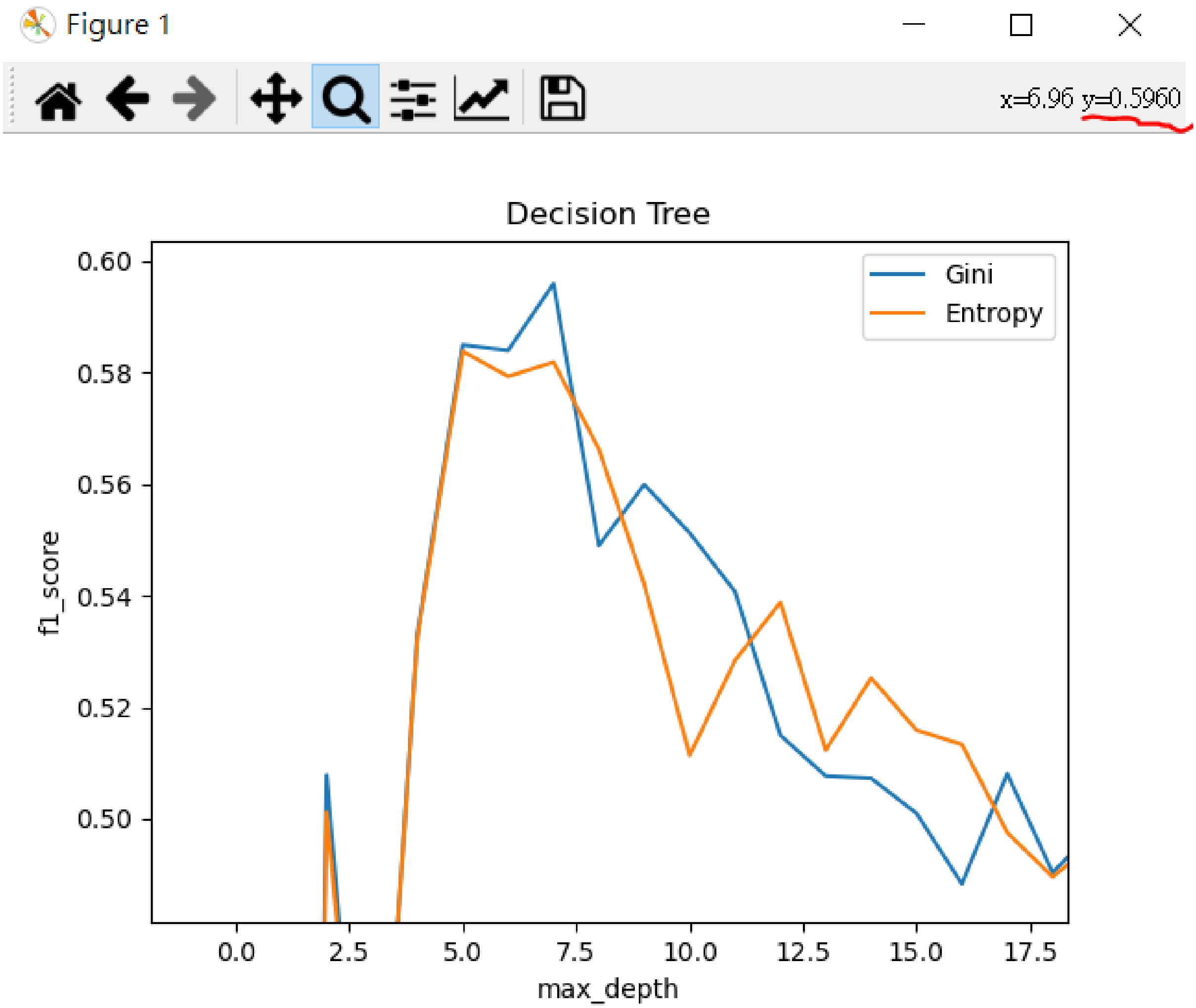
For features, we select all of them except "Surname" in our model since the decision tree model will select suitable features based on impurity score.

We run the decision tree with gini and entropy criterion for different max_depth:



When max_depth is larger than 20, "Entropy" outperforms "Gini", but what should we focus on is the peak of the graph (i.e. for max_depth around 10).

Here is the graph after zooming bigger on that region:



It shows that "Gini" is better than "Entropy" in this region.

Results:

With removing the feature "Surname" and using "Gini" with max_depth = 7, the highest f1_score can be up to 0.597.

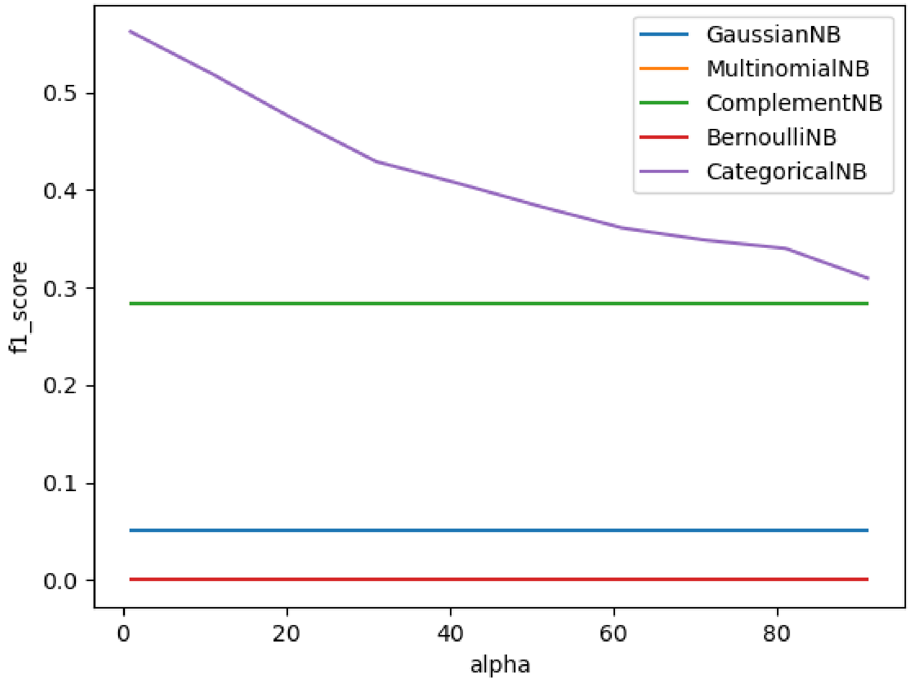
Bayes

Experimental evaluations:

In sklearn, there are 5 different Bayes classifiers. When alpha is a positive number, the Gaussian NB is the only Bayes classifier without Laplace while the others are 4 different Bayes classifiers with Laplace.

Before doing any feature selection, we use all the given attributes except "RowNumber" and "Balance" since these two attribures will generate errors when we are using the library of sklearn.naive_bayes.CategoricalNB.

Let's investigate into which NB classifier performs the best.

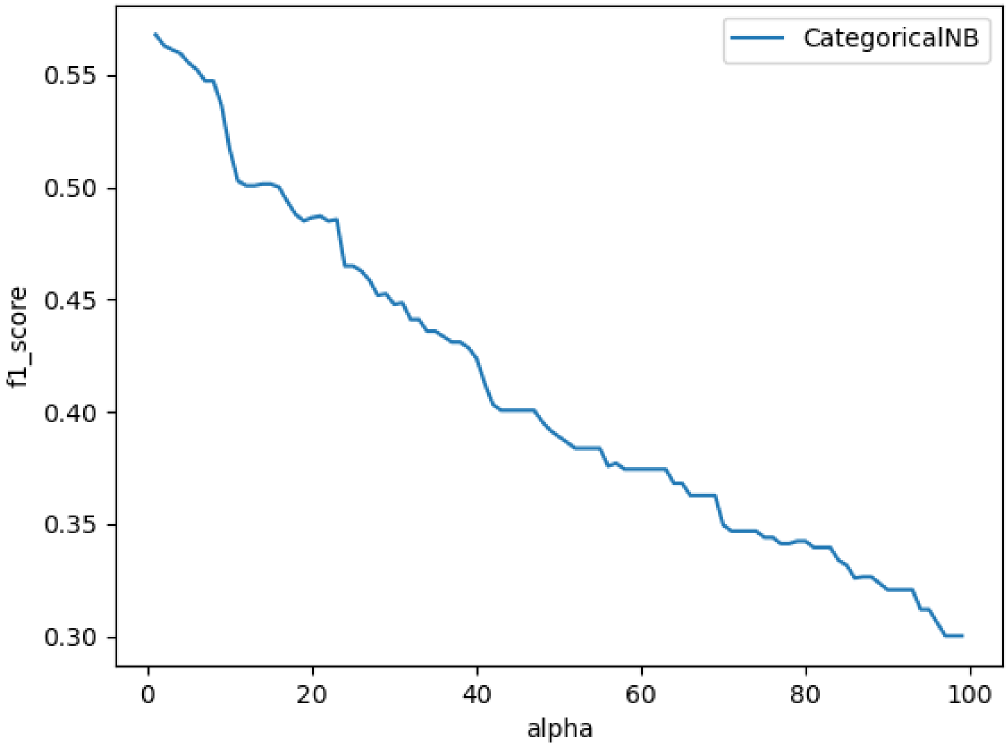


(Note: GaussianNB does not have parameter alpha.)

From the graph above, it is obvious to show that CategoricalNB performs the best with different choices of alpha. Therefore, we choose CategoricalNB (with Laplace).

For feature selection, we only choose the attributes "Geography", "Gender", "Age", "NumOfProducts" and "IsActiveMember" by human testing since without any of those attributes, it will worsen the f1_score of CategoricalNB.

Then, we tune the alpha from the range 1 to 100 to see how it will affect the f1_score of Categorical NB (see the graph below):



From the graph above, when alpha = 1, CategoricalNB performs the best.

Results:

When we use CategoricalNB (with Laplace) , select features "Geography" , "Gender" , "Age" , "NumOfProducts" and "IsActiveMember" , and choose alpha = 1, the corresponding f1_score is 0.5678704856787049.

EnsembleMethod

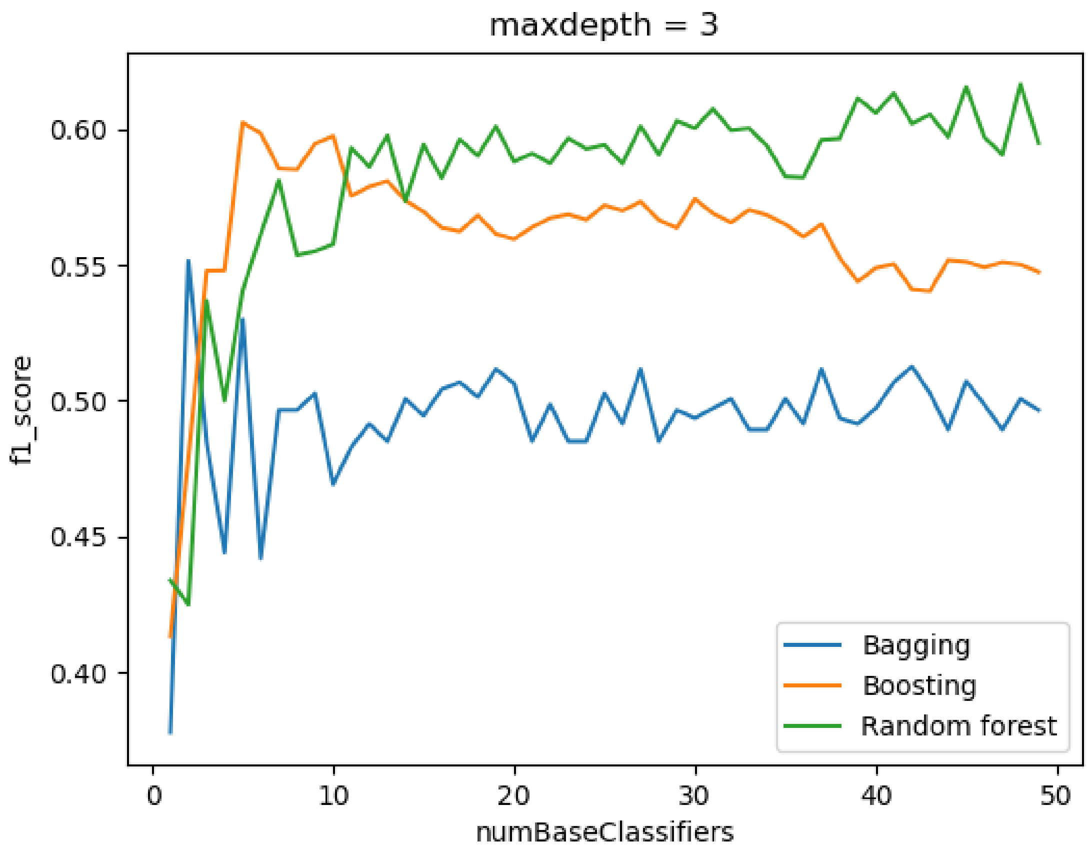
Experimental evaluations:

We use preprocess.scale as a data preprocessing method to scale down the data.

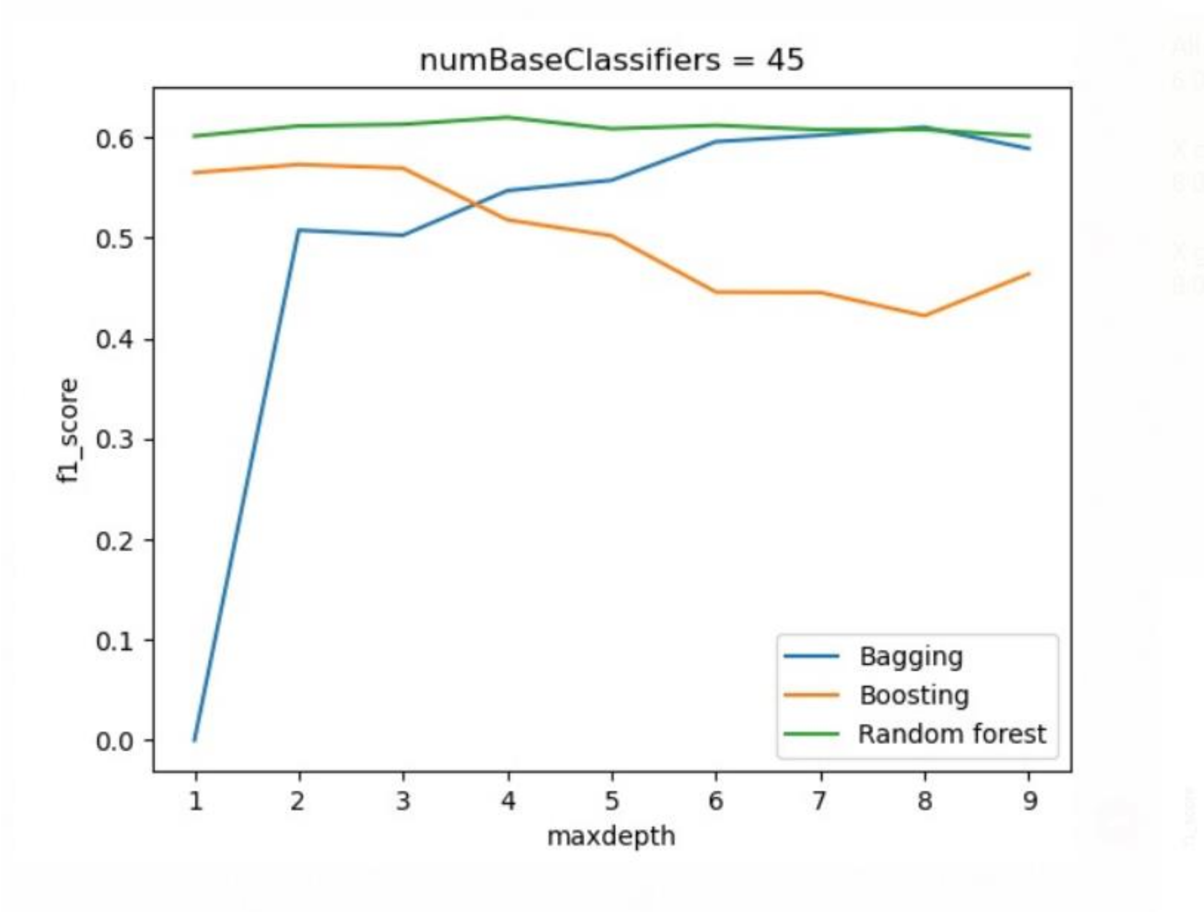
For feature selection, we choose all the features but exclude “CustomerID” and “Surname” since they are irrelevant features.

Let’s find the optimal numBaseClassifiers and maxdepth one by one.

Suppose when we fix maxdepth at random value, we can see that Random Forest performs the best when numBaseClassifiers = 45.



Now, when numBaseClassifiers = 45, with maxdepth in range 1 to 10,
Random forest performs the best when maxdepth = 4.



Results:

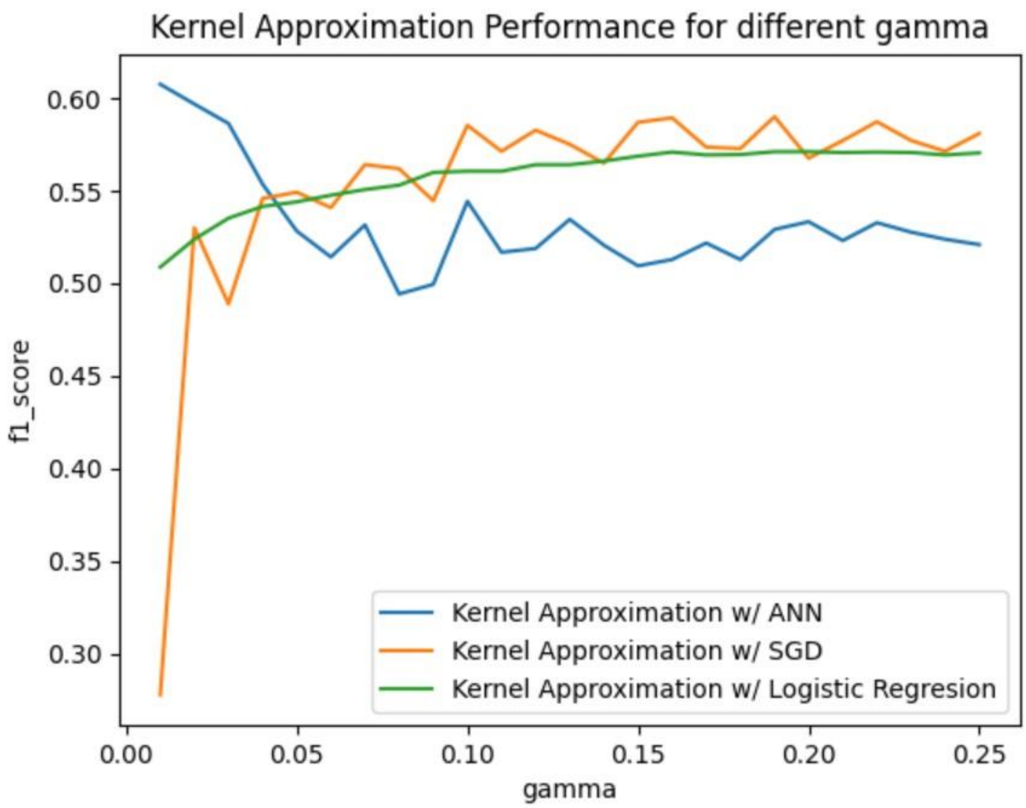
When we use preprocessing.scale to preprocess the data, select all features except "CustomerID" and "Surname", choose "Random Forest" with maxdepth = 4 and numBaseClassifiers = 45, the f1_score can be 0.62.

Kernel Approximation

Experimental evaluations:

We apply the kernel approximation technique on different algorithms and test their f1-score with different gamma, the result is shown below.

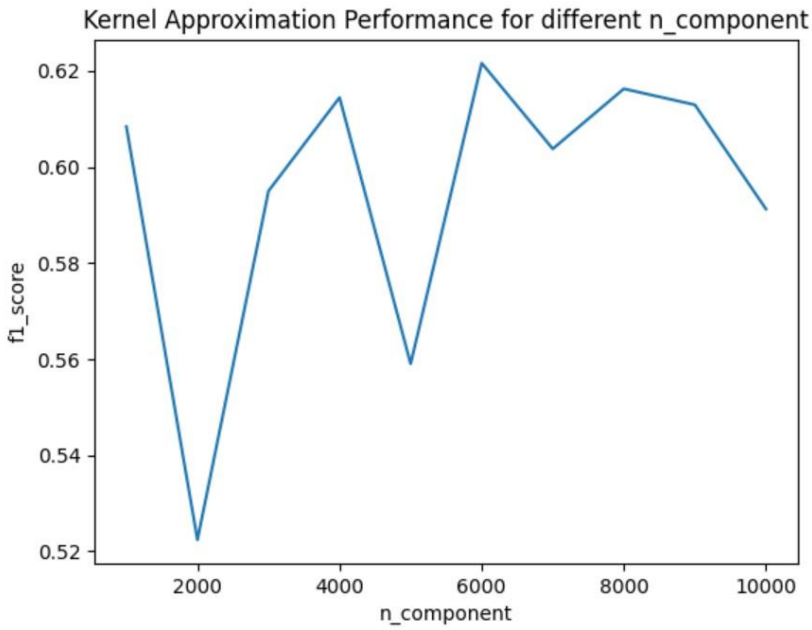
Parameters: n_components: 10000
ANN: MLPClassifier(hidden_layer_sizes=(10,10,10), solver='adam', activation= 'tanh', max_iter=5000, random_state=1)
SGD: SGDClassifier(max_iter = 1000, random_state=1)
Logistic Regression: LogisticRegression(class_weight='balanced', C=1, max_iter=5000, random_state=1)



From the graph above, Kernel Approximation with ANN performs the best when gamma = 0.01, reaching a f1-score of 0.6076555023923444.

We do further testing on n_components and the result is as follows.

Parameters: gamma = 0.01
Classifier: MLPClassifier(hidden_layer_sizes=(10,10,10),solver='adam', activation=\'tanh', max_iter=5000, random_state=1)



We can observe that when n_component = 6000, ANN with Kernel Approximation performs the best, achieving f1-score of 0.6216216216216216.

Results:

When we use preprocessing.scale to preprocess the data, select all features except "CustomerID" and "Surname". And then, we used MLP classifier with hidden_layer_sizes=(10,10,10), solver='adam', activation='tanh', max_iter=5000 and Kernel Approximation with gamma = 0.01, n_component = 6000, the f1-score can be **0.6216**.

XGBoost

Experimental evaluations:

As XGBoost runs really fast, we can first run through all combinations with default parameters and get the combinations with high f1-score. We have run 4 tests:

Original data

0.6260869565217393	['CreditScore', 'Geography', 'Gender', 'Age', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember']
0.62266500622665	['CreditScore', 'Geography', 'Gender', 'Age', 'Balance', 'NumOfProducts', 'IsActiveMember']
0.6180469715698395	['RowNumber', 'CreditScore', 'Geography', 'Age', 'Balance', 'NumOfProducts', 'IsActiveMember', 'EstimatedSalary']
0.6177215189873418	['RowNumber', 'CustomerId', 'CreditScore', 'Geography', 'Age', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember']
0.6176836861768369	['RowNumber', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember']

With preprocessing.scale

0.6277915632754343	['CreditScore', 'Geography', 'Gender', 'Age', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember']
0.62266500622665	['CreditScore', 'Geography', 'Gender', 'Age', 'Balance', 'NumOfProducts', 'IsActiveMember']
0.6173149309912169	['RowNumber', 'CustomerId', 'CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'IsActiveMember']
0.6173149309912169	['RowNumber', 'CustomerId', 'CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember']
0.6169154228855722	['CustomerId', 'Geography', 'Gender', 'Age', 'Balance', 'NumOfProducts', 'IsActiveMember']

With preprocessing.scale and PCA(n_components=6)

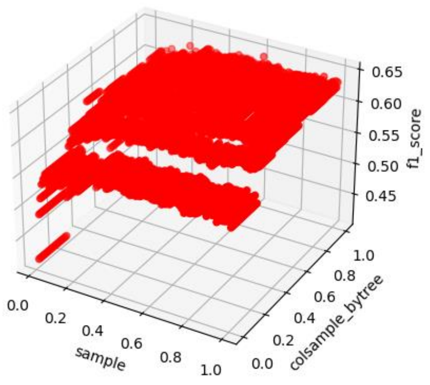
0.5414507772020725	['Gender', 'Age', 'Balance', 'NumOfProducts', 'IsActiveMember', 'EstimatedSalary']
0.5395683453237411	['Geography', 'Age', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember']
0.5321336760925449	['Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'IsActiveMember']
0.5301507537688442	['CreditScore', 'Gender', 'Age', 'Balance', 'NumOfProducts', 'IsActiveMember']
0.5260545905707196	['Geography', 'Gender', 'Age', 'Balance', 'NumOfProducts', 'IsActiveMember']

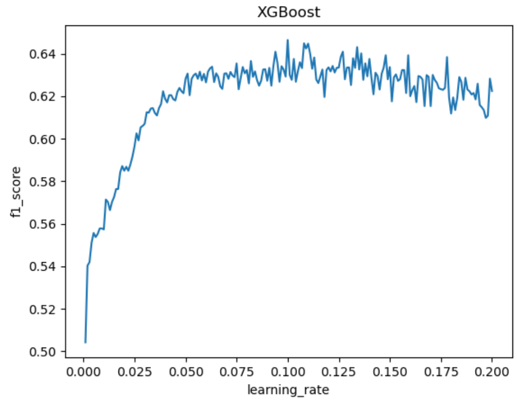
With preprocessing.scale and PCA(n_components=8)

0.5260347129506008	['CreditScore', 'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'IsActiveMember', 'EstimatedSalary']
0.5191040843214756	['CustomerId', 'Gender', 'Age', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary']
0.5126162018592298	['CreditScore', 'Gender', 'Age', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary']
0.5112582781456954	['CustomerId', 'CreditScore', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember']
0.5078125000000000	['RowNumber', 'CustomerId', 'CreditScore', 'Age', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember']

We can see that using PCA will reduce the f1-score, so we do not consider PCA. Moreover, the combination ['CreditScore', 'Geography', 'Gender', 'Age', 'Balance', 'NumOfProducts', 'HasCrCard', 'IsActiveMember'] achieved the highest f1-score both in original data and data with preprocessing.scale. Hence, we use this combination for further testing.

First, we test for data with preprocessing.scale which has the highest f1-score. We tune the parameters one by one, in sequence of n_estimators → param_max_depth, min_child_weight → gamma → subsample, colsample_bytree → reg_alpha → reg_lambda → learning_rate.





The results as follows:

Parameter(s)	Result	f1-score
(Original)	---	0.6277915632754343
n_estimators	217	0.6341463414634146
param_max_depth, min_child_weight	3, 1	0.6341463414634146
gamma	0	0.6341463414634146
subsample, colsample_bytree	0.62, 1	0.6439854191980559
reg_alpha	0	0.6439854191980559
reg_lambda	0.6	0.6464891041162227
learning_rate	0.1	0.6464891041162227

Resulting f1-score: 0.6464891041162227

We do the same things on data without preprocessing.scale (original data):

Parameter(s)	Result	f1-score
(Original)	---	0.6260869565217393
n_estimators	364	0.6308068459657702
param_max_depth, min_child_weight	3, 4	0.6359223300970875
gamma	0.491	0.6373626373626373
subsample, colsample_bytree	0.77, 0.63	0.6397058823529411
reg_alpha	0.339	0.642156862745098
reg_lambda	1	0.642156862745098
learning_rate	0.1	0.642156862745098

Resulting f1-score: 0.642156862745098

Results:

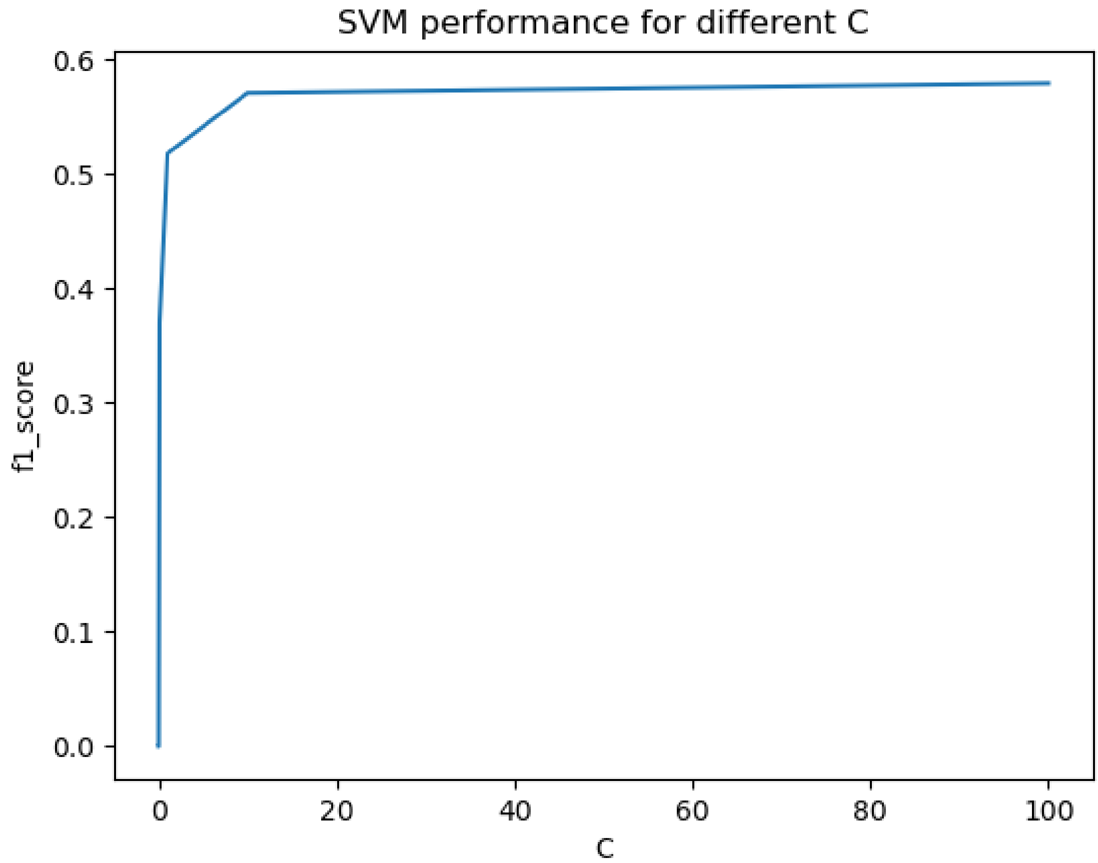
For data preprocessing, we use preprocessing.scale to scale down the dataset. With selecting features 'CreditScore', 'Geography', 'Gender', 'Age', 'Balance', 'NumOfProducts', 'HasCrCard' and 'IsActiveMember', and using the optimal parameters, which are n_estimators=217, param_max_depth=3, min_child_weight=1, gamma=0, subsample=0.62, colsample_bytree=1, reg_alpha=0, reg_lambda=0.6 and learning_rate=0.1, f1-score can be 0.6464891041162227.

SVM

Experimental evaluations:

Preprocessing method: preprocessing.scale

Feature selection: Use all features without "Surname



Parameter choice: C=100, kernel='poly'

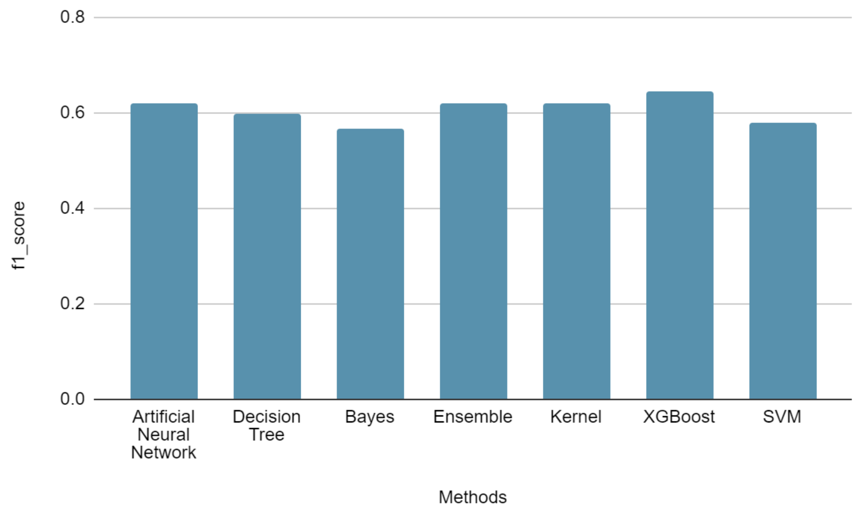
Results:

When we use preprocessing.scale to preprocess the data, select all features except "Surname" . And then, we choose C=100 and 'poly' for kernel,

f1_score: 0.5790139064475348.

Conclusions:

Methods	f1_score
Artificial Neural Network	0.6216216216216216
Decision Tree	0.5972850678733032
Bayes	0.5678704856787
Ensemble	0.6217870257037944
Kernel	0.6216216216216216
XGBoost	0.6464891041162227
SVM	0.5790139064475348



We have tried several methods with trying different data preprocessing methods, sel ecting different combinations of features, and tuning the parameters to maximize the f1-score of the models. All of them have the f1-score larger than 0.56, and the most out-standing one is XGBoost. Its f1-score can be **0.6464891041162227**.