CS 580: Artificial Intelligence (P3)

Md. Ridwan Hossain Talukder

I. Analysis of Bank Marketing Dataset

For the first part of the project we are using a dataset that is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (variable y). In this project I have implemented a k-nearest neighbors classifier, a decision tree classifier, a random forest classifier, and another classifier of your choice to predict the variable y.

A. Data Preprocessing

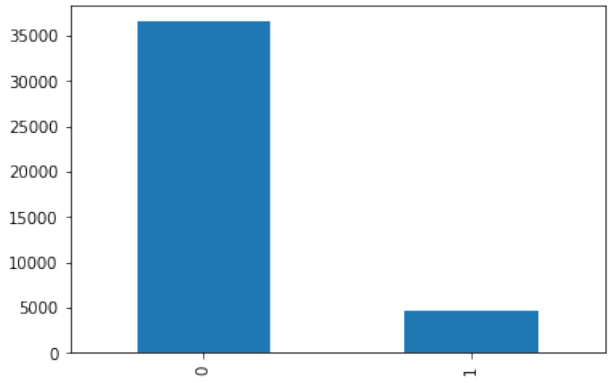

Fig. 1: Imbalanced distribution of dataset

The dataset originally contains 41188 record with 21 features. It has 10 categorical features and 11 numeric features. I first checked whether there is any null or NAN values in any columns, but there were no records with a NaN or null values. Next I checked for the duplicate records and there were 12 duplicate records in the data. I removed those record. As I do not have a test data to validate whether the model works good on unknown data set so I splitted the original data into 3:1 ratio (30882 training data and 10294 test data) while keeping the target (yes:no) label ratio same. I Kept the test data for validation after tuning the models so that the test data remains new. Next I applied one hot encoding technique for all the categorical features using the pandas get_dummies method as most ML algorithm require numerical inputs. I did not use label encoding technique as it may create unreal relationship between categorical features by ordering them and this may contribute to incorrect prediction. Now we encode the target(y) labels to 0 and 1, where 0 is No and 1 is Yes. We count the target values in training set and found that it has only 4639 'Yes' target values comparing to 36537 'No' target values. So this dataset has an imbalanced target dataset, and while training the models we have to keep that in mind. Also as this is an imbalanced dataset, accuracy may not be a good performance measurement, because if we predict all the target as 0, we may still get a higher accuracy rate but it may not tell us how good our model would be predicting any new data. So we will also have to consider f1 score along with accuracy as our performance measurement metric.

B. k-nearest neighbors classifier

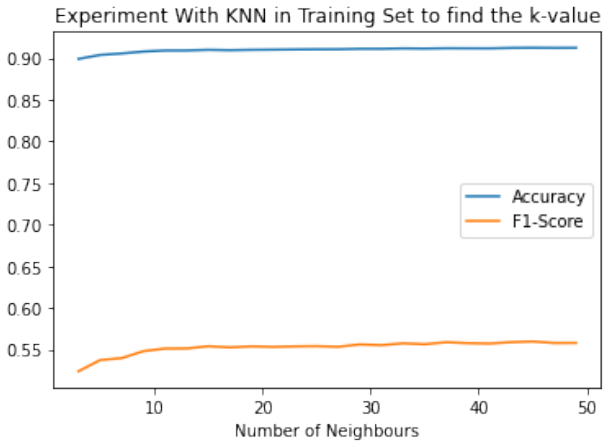

Fig. 2: Experiment on Training Dataset (KNN) to find the best K value

1) Experiment on Training Dataset: So we got our training dataset by splitting the original dataset into 3:1. Now in this training dataset we ran our first experiment to get the k-value for which we get the highest accuracy. We used K fold Stratified Cross validation method as this is an imbalanced dataset and Stratification is needed to have the balanced ratio of training and testing data while doing the cross validation. We used 5-fold and 1-repeat for the cross validation and tested it for a neighbour of 3 to 50 and found out that when the value of k is near 33 the accuracy is highest and it does not improve much by increasing the k value. But the f1 score is not that high but we also get the highest f1 score when the k value reaches to this value (33). While getting a higher accuracy seems good but this does not mean that our model will perform good on the test data. We used weighted voting and used euclidean distance as the distance metric for the knn classifier.

As we did experiment changing the distance metric and voting criteria we found out that majority voting and weighted voting produces identical results while euclidean distance metric produces the best result for the KNN.

2) Prediction on Testing Dataset: So, as we chose K=33 and Euclidean Distance as the distance Metric, and weighted voting amongst the neighbours. We now predict the labels on the test data set using those parameters in our K-nearest

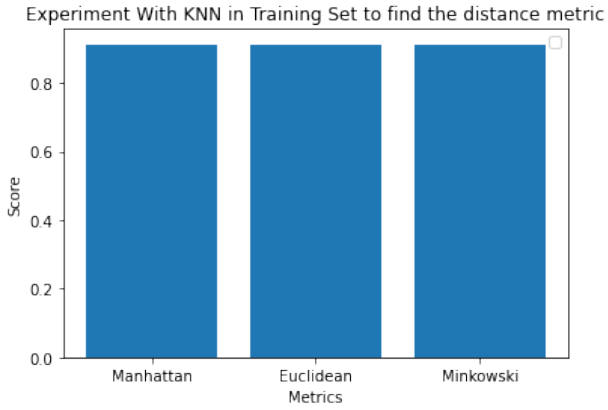

Fig. 3: Experiment on Training Dataset (KNN) to find the best distance metric

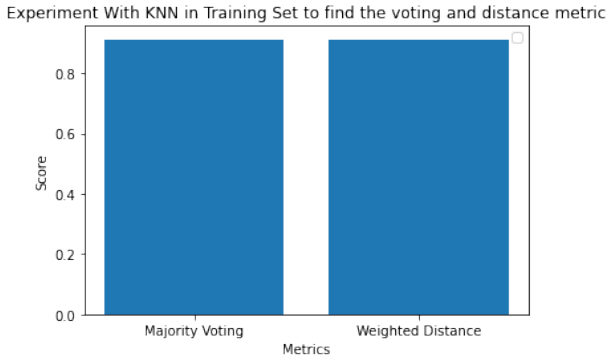

Fig. 4: Experiment on Training Dataset (KNN) to find the best voting technique

neighbour classifier and we got an accuracy of 91% in the test dataset as well while the f1_score is 0.54 which is still low.

C. Decision Tree classifier and Random Forest Classifier

I used the Decision Tree Classifier and Random Forest Classifier with the default parameters to predict on the Test data set and without changing the parameter much I got 88.89% accuracy on Test Data for Decision Tree Classifier and 90.9% accuracy for Random Forest Classifier. While the f1 score was still low (0.50 for Decision Tree Classifier and 0.42 for Random Forest Classifier)

D. Other Classifier

The other classifier I used is the Adaboost Classifier with Decision Stump (Decision Tree with Max depth 1) as the base estimator. I first re-sample the training data set using the random under sampling method and then use the AdaBooster for classification and I got 87% accuracy on Test data set but the F1 score is the Highest and it is 0.60 which is very high from the other classifiers. As this is an imbalanced dataset, f1 score metric is more important to determine model's performance than accuracy as most of the classifiers will get a higher accuracy on the data set but f1 score may be still low.

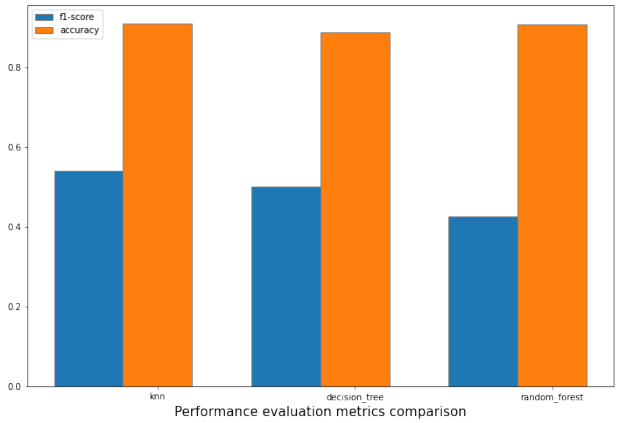

Fig. 5: Experiment on Test Dataset using all three classifiers

So AdaBoosterClassifier outperforms those three classifiers in terms of F1 score metric.

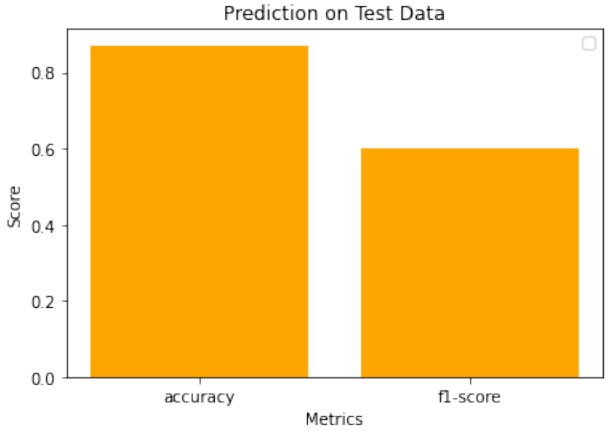

Fig. 6: Experiment on Test Dataset using AdaBoostClassifier

E. Results

TABLE I: F1-Score and Accuracy for all the 4 models

Metric	KNN	DT	RF	AB
f1_score	0.54	0.50	0.42	0.60
accuracy	0.91	0.88	0.90	0.87

II. Prediction of Motion-Planning Distance

The objective of this second part of the project is to predict the solution distance found by Bug2 when given the start position and the goal position using k nearest regressor, decision tree regressor, random forest regressor and another regressor of my choice. In this problem we use a bug2 data set with only 5 columns (sx, sy, gx, gy and d) where (sx, sy) is the start position and (gx, gy) is the goal position and d is the distance. There are 100000 instances with 5 columns in the dataset. As we do not have any other testing dataset so we split the data into 3:1 ratio and keep the testing data aside from training to make a good prediction later. I used r^2 value

to calculate the performance of the regressor. r^2 value is the coefficient of determination when the value is close to 1.0, it is considered to be a good regression model.

A. Data Preprocessing

First we check if there is any NaN or Null value in the dataset, and remove them. As there were no such value we did not perform such operation on the dataset. Then we checked if there were some duplicate values and we found 1, so removed that data to remove unnecessary data points. As the variables (sx, sy, gx, gy) are continuous, and has a different value from different range, we need to scale all the values to (0 to 1). We used the MinMaxScaler from sllearn.preprocessing library for that purpose which transforms all the features by scaling each feature to a given range.

B. Regressors

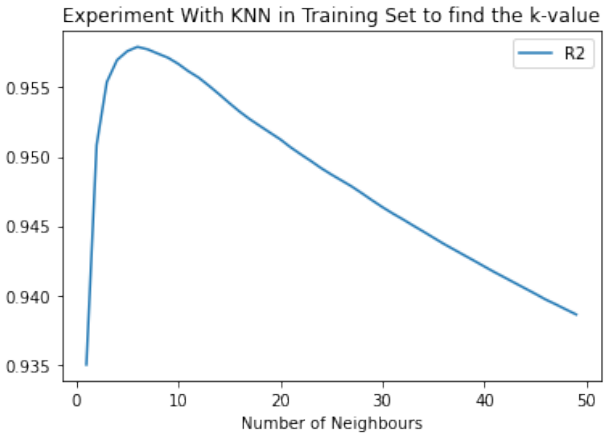

Fig. 7: Experiment on Training Dataset (KNN Regressor) to find the best k value

1) KNN: For KNN, we used the KNeighborsRegressor. To determine the best value of k for the regressor we ran the experiment with k-fold (k=5) cross validation. We choose the values from 1 to 50 and in the graph we can see that for k = 6 we got the highest r^2 value, where the curve is at it's peak and then it started to come down again. So we picked the k value as 6 for the KNeighborsRegressor. The we ran the experiment to find the best distance metric for KNN and we got the same value for all the metric so we choose euclidean as the default distance metric.

2) Decision Tree Regressor and Random Forest Regressor: I used the Decision Tree Regressor and Random Forest Regressor with the default parameters to predict on the Test data set and without changing the parameter much I got 0.906 (r^2) on Test Data for Decision Tree Regressor and 0.956 (r^2) accuracy for Random Forest Regressor. So I did not do any more experiments of parameter tuning rather used the default values for the regressors.

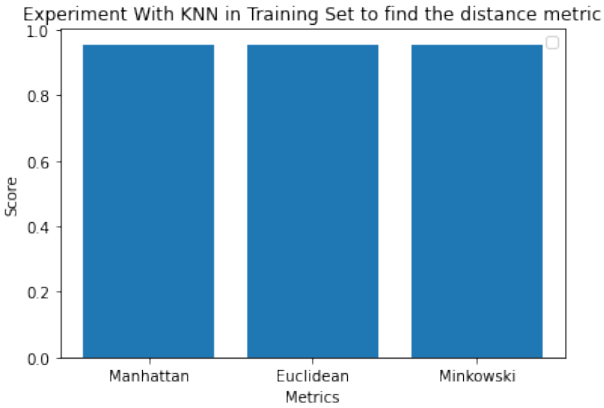

Fig. 8: Experiment on Training Dataset (KNN Regressor) to find the best distance metric

3) VotingRegressor: As the motivation to use the other regressor is to improve the prediction, it became really a very hard job considering that already knn regressor did a pretty good job on the test data set (achieving 0.960012 r^2 value). So I used a voting regressor which is an ensemble meta-estimator that fits several base regressors, each on the whole dataset. Then it averages the individual predictions to form a final prediction. For my voting regressor I used the Knn Regressor and Random Forest Regressor and achieved a r^2 value of 0.962468 which is slightly better than the Knn alone.

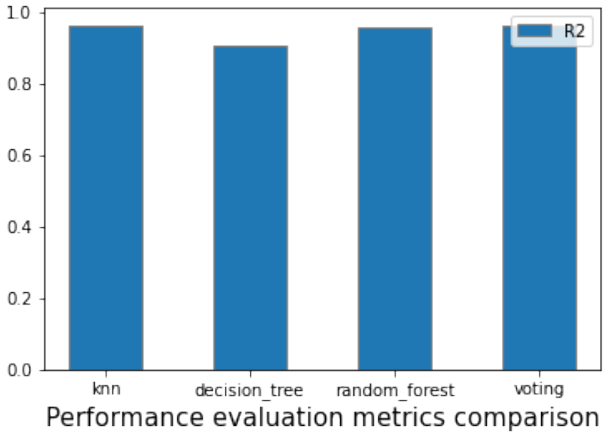

Fig. 9: Experiment on Test Dataset for all 4 regressors

C. Results

TABLE II: R2-Score for all the 4 models

Metric	KNN	DT	RF	VR
r2_score	0.960	0.90	0.95	0.9624