SAPIENZA
UNIVERSITÀ DI ROMA

Department of Computer, Automatic and Management Engineering

Machine Learning HW01
Classification and Regression models

Pouya Irani
Matricola: 1937388
Email: irani.1937388@studenti.uniroma1.it

Table of Contents

1 Dataset description

The dataset is included 5 UAVs which follow a defined path, and they might collide to each other, and the experiment has been done 1000 times with different starting and finishing position and with various speeds as well.

2 Classification Problem

The problem of classification is to obtain and predict the number of collisions in the experiment.

2.1 Preprocessing

Before performing any pre-processing model, we need to analyze the dataset to realize the number of classes and data distribution.
Thou the classes are as follows:

```
0    538
1    333
2     96
3     30
4      3
Name: num_collisions, dtype: int64
```

As it clearly visible the classes are imbalanced and if the classification be performed now the accuracy would be very low hence with the train and test set split command, there is a high probability to lose class 3 and 4 in either training or testing phase.

2.2 Over sampling

One approach to addressing the problem of class imbalance is to randomly resample the training dataset. The two main approaches to randomly resampling an imbalanced dataset are to delete examples from the majority class, called undersampling, and to duplicate examples from the minority class, called oversampling.

2.2.1 SMOTE

SMOTE stands for Synthetic Minority Oversampling Technique.
SMOTE is an algorithm that performs data augmentation by creating synthetic data points based on the original data points. SMOTE can be seen as an advanced version of oversampling, or as a specific algorithm for data augmentation. The advantage of SMOTE is that you are not generating duplicates, but rather creating synthetic data points that are slightly differentfrom the original data points.
After SMOTE application on the dataset the class distributions is as follows:

```
0    538
1    538
2    538
3    538
4    538
dtype: int64
```

2.3 Grid search

Grid-search is used to find the optimal hyperparameters of a model which results in the most 'accurate' predictions.
In this project I did the Grid search to find the best number of neighbors to perform K-nearest neighbors' model.
It can be performed simply by using sklearn python package.

```
#Grid search to find the best parameters for KNN

from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier

clf = KNeighborsClassifier()
k_range = list(range(1, 100))
param_grid = dict(n_neighbors=k_range)

# defining parameter range
grid = GridSearchCV(clf, param_grid, cv=10, scoring='accuracy', return_train_score=False,verbose=1)

# fitting the model for grid search
grid_search=grid.fit(x_smote, y_smote)
```

```
Fitting 10 folds for each of 99 candidates, totalling 990 fits
```

```
print(grid_search.best_params_)
```

```
{'n_neighbors': 1}
```

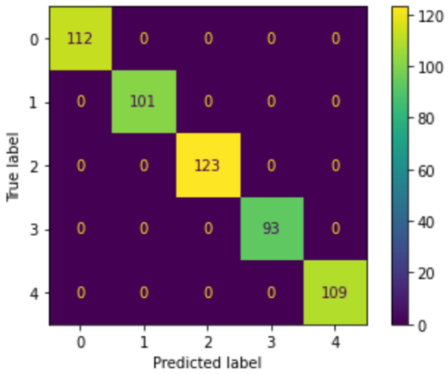
Since it has shown above the best n_neighbors are 1.

2.4 KNN

In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

```
#Fitting the KNN classifier

from sklearn.metrics import accuracy_score, plot_confusion_matrix

knn = KNeighborsClassifier(n_neighbors=1)

knn.fit(x_train, y_train)

y_test_hat=knn.predict(x_test)

test_accuracy=accuracy_score(y_test,y_test_hat)*100

print("Accuracy for our testing dataset with tuning is : {:.2f}%".format(test_accuracy) )
```

```
Accuracy for our testing dataset with tuning is : 82.53%
```



The model is performing almost accurate as it can be realized from above figures both in accuracy score and confusion matrix.

2.5 SVM

The objective of the support vector machine is to find a hyperplane in an N-dimensional space (N- the number of features) that distinctly classifies the data points.
To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

I have examined 2 different kernels for SVM to see the effect of them on accuracy and F-1 score and as it has shown below the RBF kernel performed slightly better for this dataset.

```
#Executing SVM

clf = svm.SVC(kernel='rbf', degree=3, C=1)
clf.fit(x_train ,y_train)

accuracy = clf.score(x_test , y_test)
f1 = f1_score(y_test, poly_pred, average='weighted')
print('Accuracy (rbf Kernel): ', "%.2f" % (accuracy*100))
print('F1 (rbf Kernel): ', "%.2f" % (f1*100))
```

```
Accuracy (rbf Kernel):  84.01
F1 (rbf Kernel):  80.33
```

```
poly = svm.SVC(kernel='poly', degree=3, C=1).fit(x_train, y_train)
```

```
poly_pred = poly.predict(x_test)
```

```
from sklearn.metrics import f1_score
poly_accuracy = accuracy_score(y_test, poly_pred)
poly_f1 = f1_score(y_test, poly_pred, average='weighted')
print('Accuracy (Polynomial Kernel): ', "%.2f" % (poly_accuracy*100))
print('F1 (Polynomial Kernel): ', "%.2f" % (poly_f1*100))
```

```
Accuracy (Polynomial Kernel):  81.41
F1 (Polynomial Kernel):  80.33
```

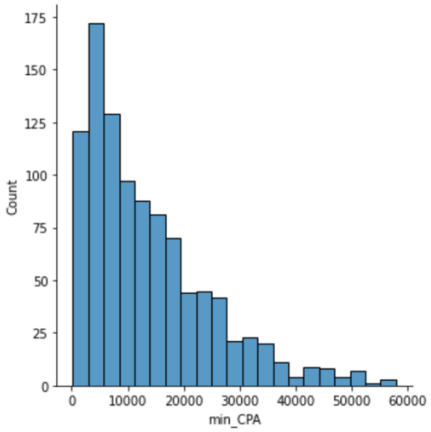
3 Regression Problem

The goal of this task is to predict the closest distance of approach in each experiment of the dataset

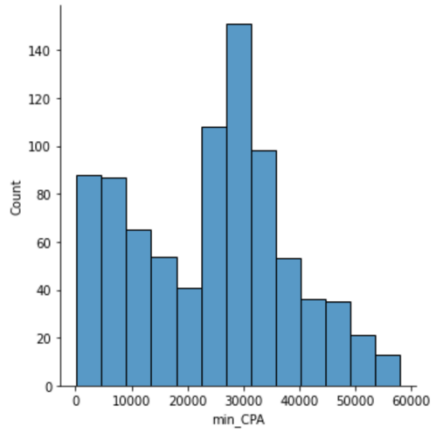
3.1 Preprocessing (SMOGN)

A Python implementation of Synthetic Minority Over-Sampling Technique for Regression with Gaussian Noise (SMOGN). Conducts the Synthetic Minority Over-Sampling Technique for Regression (SMOTER) with traditional interpolation, as well as with the introduction of Gaussian Noise (SMOTER-GN). Selects between the two over-sampling techniques by the KNN distances underlying a given observation. If the distance is close enough, SMOTER is applied. If too far away, SMOTER-GN is applied. Useful for prediction problems where regression is applicable, but the values in the interest of predicting are rare or uncommon. This can also serve as a useful alternative to log transforming a skewed response variable, especially if generating synthetic data is also of interest.

The comparison between the original data and processed data with the mentioned method has visualized below.



1 original data



2 Cleaned data with SMOGN

3.2 Regression algorithms

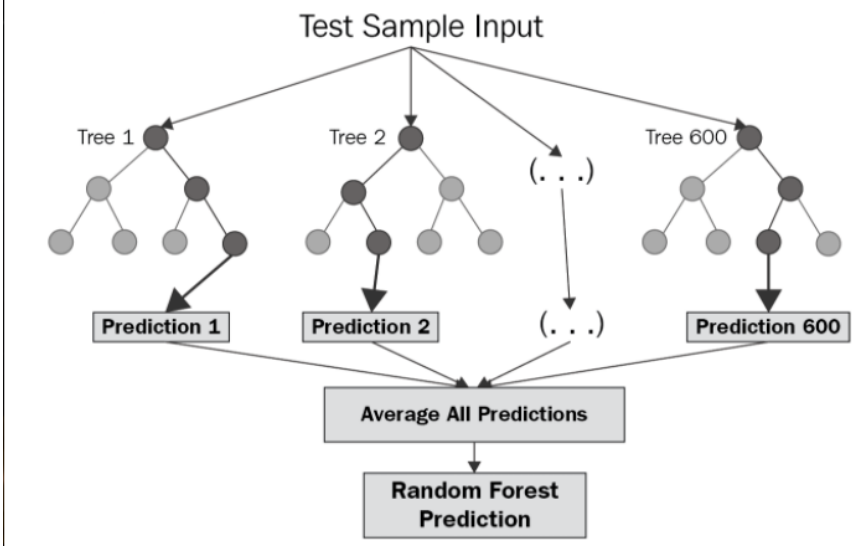
3.2.1 Linear Regression

Linear regression analysis is used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.

This form of analysis estimates the coefficients of the linear equation, involving one or more independent variables that best predict the value of the dependent variable. Linear regression fits a straight line or surface that minimizes the discrepancies between predicted and actual output values. There are simple linear regression calculators that use a “least squares” method to discover the best-fit line for a set of paired data. You then estimate the value of X (dependent variable) from Y (independent variable).

3.2.2 Random forest

Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.



The diagram above shows the structure of a Random Forest. You can notice that the trees run in parallel with no interaction amongst them. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees.

3.2.3 Polynominal Regression

Polynomial regression is a form of Linear regression where only due to the Non-linear relationship between dependent and independent variables we add some polynomial terms to linear regression to convert it into Polynomial regression.

Suppose we have X as Independent data and Y as dependent data. Before feeding data to a mode in preprocessing stage we convert the input variables into polynomial terms using some degree.

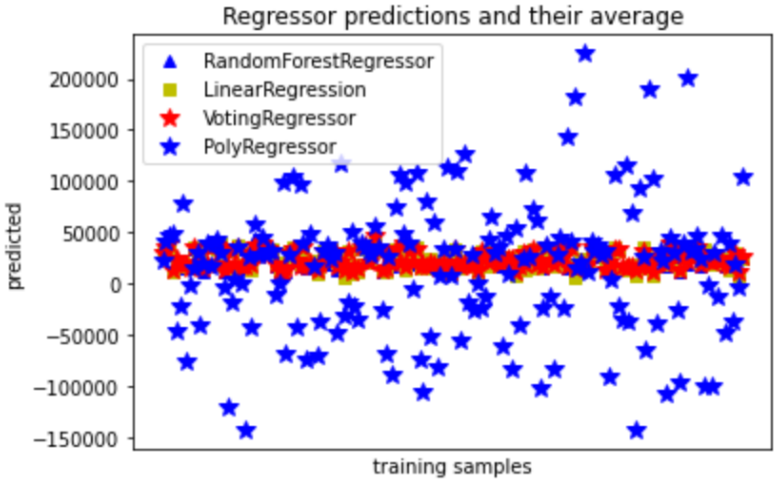
3.2.4 Voting Regressor

A voting regressor 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.

3.2.5 Results

Below has shown the comparison between all regression models that introduced

```
RF Mean squared error: 106449515.89
RF Regression score: 0.41

LR Mean squared error: 139215614.72
LR Regression score: 0.23

V Mean squared error: 115438543.74
V Regression score: 0.36

poly Mean squared error: 115438543.74
poly Regression score: 0.36
```



Since the parameters of dataset is distributed massively and cannot be preprocessed (estimated and limited in range with e.g minmax scaler) because the parameters are positions and speed and logically by scaling the loss of actual information would be inevitable, the accuracy of the shallow and common regression models are low.
Probably with the neural network high performance models (Which is out of scope of this homework) like Reinforcements learning which seems handy for this dataset and this specific task the results can be much more accurate.