MACHINE LEARNING

PROJECT REPORT

SEMESTER 1, 2020/2021

CSC 3304, SECTION: 01

LECTURER: DR. AMIR 'AATIEFF

Prepared by:

Name	Matric number
MD RAKIBUL HASSAN	1720465
MUHAMMAD HAZIQ ADLI BIN ZAMZURI	1814981
TANVEER MAHMOOD HASAN	1725413
FADHLUDDIN BIN SAHLAN	1817445
YASIN MOHAMMED	1814111

1. INTRODUCTION

BACKGROUND

This dataset is provided by the City of Austin's Open Data Initiative and details the Austin Animal Center's (AAC) shelter pets and their respective outcomes. The goal of the Austin Animal Center is to provide long-term, sustainable outcomes to the animals in their care, through adoption, foster care, or other rescue shelters. At the time of writing, they shelter around 16,000 animals annually, regardless of age, health, species, or breed.

The original dataset titled ***"aac_shelter_outcomes"*** lists the details of all the species under the AAC's care, which includes cats, dogs, and even some bats. However, a more refined dataset titled ***"aac_shelter_cat_outcome_eng"*** containing only the details of cats will be used for the purposes of this assignment, courtesy of Aaron Schlegel. This dataset contains 37 columns (2 of which are our target variables), and 29,421 records (29,418 records after preprocessing). More details about the data will be elaborated on in the methodology section of the report.

OBJECTIVE

The objective of this project is to predict the outcome of each animal based on the features at our disposal. The outcome is what ends up happening to the pet, be it adoption, return to owner, or death. Machine learning is employed to solve this problem as it will be most effective in figuring out which parameters have an effect when it comes to the outcome.

The rationale of the objective is simple. If there are any visible patterns or trends derived as a result of our investigations, it will help optimize for the outcome that is preferred, i.e. adoption. Certain parameters are within the AAC's control, so the knowledge obtained will allow them to curate those parameters. On the other hand, the parameters not within the AAC's control will hopefully assist them in directing resources to the right place.

A higher-level objective is to utilize additional techniques beyond the simple application of an ensemble method. This is due to the fact that this project is built upon an earlier project (ML individual assignment: 1817209). For instance, better feature engineering, or hyperparameter tuning. This can be further explored later in the report.

2. METHODOLOGY

INTRODUCTION TO THE RANDOM FOREST ALGORITHM

In this project we would like to use the Random Forest approach which is quite straightforward. It is a supervised learning algorithm which uses the approach of ensemble learning for classification and regression. There is no inter relation between these trees while constructing the trees. It works by building a multitude of decision trees at training time and giving output which is a mode of classification and regression. It can combine the multiple predictions. We have decided to use the Random forest method because compared to other algorithms Random Forest gives more accuracy.

ADVANTAGES WHEN COMPARED TO DECISION TREES

Decision trees are sensitive to work with our dataset. In our case, the results of the decision tree can be quite different if the training data is altered and the predictions can also be slightly different in turn. In addition, it is also costly to train computationally, bear a great risk of overfitting, and appear to find local optima because after splitting it cannot be undone. (Chakure 2019 P.4).

Decision tree works on a whole dataset that uses all the columns or features. On the contrary, Random forest works based on randomly selected observations and specific columns or features to build numerous decision trees and then make a conclusion with the averages of results.

BENEFITS OF THE RANDOM FOREST ALGORITHM

According to Julia Kho (2018), Random forest is suitable for any requirement whether the model is regression or classification tasks. It is capable of handling numerical functions, binary features or categorical data. There seems to be little pre-processing that must be conducted. There is no need to transform or rescale the data.

Furthermore, Decision tree is considered a weak learner since it cannot make great predictions on unknown samples even though it does very well on training data (over-fitting). So, by combining multiple trees into 'forests', the Random forest algorithm will enhance the predictions of our model. This method is called Ensemble methods which use multiple learners or algorithms and group them together to create a stronger machine learning model. For Random Forests, they are an ensemble of multiple single Decision Trees.

Lastly, every single Decision Tree contains a low bias (training error) with high variance (test error). So, Random Forests algorithm will average all Decision Trees in the model as well as averaging the variance resulting in a low bias and moderate variance machine learning model.

CHALLENGES OF USING THE RANDOM FOREST ALGORITHM

Formally, the terminologies of the Random Forest Regression are as follows:

1. The number of features which one is selected that can be split on at each node is limited to some percentage of the whole.

2. When splitting, each tree is drawn from the original dataset by random sampling, introducing the next element of randomness that avoids overfitting.

3. Major challenge is that Random Forest can not extrapolate outside unseen data. On the contrary, to obtain non-linear relationships between input features and the target variable, Decision Trees are perfect.

4. It can not perform well when data is in the form of a time series. Problems with time series include finding an increasing or decreasing pattern that would not be able to be formulated by a Random Forest Regression.

FEATURE ENGINEERING: PRINCIPAL COMPONENT ANALYSIS

Principal Component Analysis (PCA) is a method to reduce dimensionality and it is usually used to reduce the dimensionality of big datasets. PCA converts a large set of features into smaller one without affecting the comprehensiveness of the dataset so much as the remaining features still contain the required information. Before performing PCA to the dataset, the dataset already filtered with only logically required features which are "sex", "Spay/Neuter", "outcome_age_(days)", "breed1", "breed2", "cfa_breed", "coat" and the target is “outcome_type”.

Though the number of features has been reduced, PCA might still be required to reduce it further since the number of tuples in the dataset is large. Further reducing the dimensionality can help in reducing noise in data and also reduce the computational power required to train the model. Less features to compare with means less time to train the model.

The first step of performing PCA is standardizing the data. This is to avoid any features overpower others that can cause bias to the data. There are many methods to standardize the data where in this project, z-score is used. Once the standardization is done, all the features will be on the same scale. The formula of calculating z-score is as below.

z=value-mean/standard deviation

The next step is to compute the covariance matrix which contains the covariance of all possible pairs of the features available. Once this is computed, the eigenvectors and eigenvalues of the matrix will be computed to identify the principal components. Since, initially the data has 7 features, it will output the same number of principal components and it is arranged from components with high information to the low. The principal component that has large value means it has more variance in it thus more information it has. The PCA variance ratio of the data is as below:

PC1	**PC2**	**PC3**	**PC4**	**PC5**	**PC6**	**PC7**
0.2433780	0.1661948	0.1554820	0.1418654	0.1248958	0.1147955	0.0533886

Each of the principal components have their own value of ratio which describe their variance of the information they hold. Each of them also has their feature vector that describes the importance of certain features from others. We are going to choose only the first two principal components since they are the most significant among others. The feature vectors for each of the two components are as follows:

	0	1	2	3	4	5	6
PC1	0.024695	0.068548	0.154783	0.665036	0.120127	0.667859	0.260759

PC2	0.156506	0.712900	0.658089	0.146937	0.007568	0.103377	0.043168

"Sex" = 0,

"Spay/Neuter" = 1,

"outcome_age_(days)" = 2,

"Breed1" = 3,

"Breed2" = 4,

"Cfa_breed" = 5,

"Coat" = 6

For PC1, “Breed1” and “Breed2” is the most significant features and for PC2, “Spay/Neuter” and “outcome_age_(days)” is the most significant features. Thus, these four features are selected to be the features of the model.

HYPERPARAMETER TUNING: GRIDSEARCH

To further improve the accuracy of our model, we tried tuning the hyperparameters of the random forest classifier model. The random forest classifier has various parameters that can be changed before using it for training. The important ones among them are listed below (Koehrsen, 2018):

n_estimators - This parameter represents the number of trees in the forest

max_features - This parameter defines the maximum number of features that would be considered before splitting a node

max_depth - This defines the maximum depth of each tree

min_samples_split - Used to specify the minimum number of data points placed in a node before it is split

min_samples_leaf - Used to specify the minimum number of data points in a leaf node

bootstrap - Defines the method for sampling data points

Trying different combinations of all these parameters to find the one that gives the best accuracy is not quite feasible if done manually. Instead of that, we may utilize a special method built exactly for this called Grid Search that can automate the testing of different parameters on the model and provide us with the best one from among them. Since this method involves testing multiple combinations of parameters on the same dataset, it is prone to overfitting. To avoid this, grid searching is usually done

along with cross-validation. The GridSearchCV class provided by scikit-learn has cross-validation built-in to ease the process. To run a Grid Search, we first define a range of values, the “grid”, for the parameters that are to be tested. Then we can proceed to create a grid search model with random forest classifier as our estimator. This model can then be trained on our training set using the fit() method, just like other models, upon which it will start testing all the possible combinations of values from the defined range.

In our case, we defined the following range for the parameters in our Grid Search model:

```
param_grid = {
   'bootstrap': [True],
   'max_depth': [40, 50, 60, 70],
   'max_features': [3, 4, 5],
   'min_samples_leaf': [4, 5, 6],
   'min_samples_split': [6, 8, 10],
   'n_estimators': [150, 200, 250]
}
```

3. RESULTS & DISCUSSION

FEATURE ENGINEERING: PRINCIPAL COMPONENT ANALYSIS

After training the model using the PCA derived sets of features only, the model is improved in its performance slightly by 1% to 2% rather than using all features. By using all features, the model achieved the accuracy of 79.84% during testing and 84.45% during training. By using only those four selected features, the model achieved 81.2% on testing data and 81.67% on training data. It seems like using all features without doing feature engineering might cause overfitting of the model. Though the model shows improvement, there is a doubt whether it is good to use PCA on such categorical data.

HYPERPARAMETER TUNING: GRIDSEARCH

After running Grid Search on the model with number of folds for cross validation set to 3, it found the following combination of parameters to be the best among all the ones tested:

```
{'bootstrap': True,
 'max_depth': 60,
 'max_features': 3,
 'min_samples_leaf': 6,
```

'min_samples_split': 6,

'n_estimators': 150}

Training our model with the above parameters yielded an improvement of 2% in accuracy over the base model:

Metrics from base model:

```
score on test: 0.7999660095173351
score on train: 0.8452026854763321
```

Metrics from hypertuned model:

```
score on test: 0.8205302515295717
score on train: 0.8259114472677828
```

This was tested by setting the random_state parameter of the Random Forest Classifier to a constant value of 42, so that the difference is not influenced by the randomness. This is not very significant, but nonetheless, it is still an improvement from the base model we started with. Hypertuning is usually the last attempt to squeeze out a little bit more accuracy from the model and hence, does not result in a significant increase in accuracy.

FINAL MODEL

```
score on test: 0.8173011556764106
score on train: 0.826378856123056
[[2958  377]
 [ 698 1851]]
0.8173011556764106
0.7261671243624951
0.8307899461400359
```

We will utilize the standard suite of performance metrics to evaluate our model. They are: Accuracy, sensitivity (or recall), specificity, and precision (positive predictive value).

From the above figure:

- Accuracy = 82%
- Sensitivity = 82%
- Specificity = 73%

- Precision = 83%

To provide further context to the results obtained, let's look at the implications. An accuracy of 82% is certainly quite impressive. But it does not usually tell the whole story. For that, we will need to look at the other metrics.

Sensitivity (or recall) is an indicator of how many of the actually positive cases that was flagged correctly. Precision, on the other hand, is an indicator of how many of the flagged cases were actually positive. High sensitivity tries to account for false negatives, while high precision tries to account for false positives. Specificity fulfills a similar role to precision and is often at odds with sensitivity.

As mentioned before, accuracy is not a good one-size-fits-all solution when it comes to performance metrics. The other metrics provide nuance and is highly contingent on the problem we are trying to solve. False positives do not always hold the same weight as a false negative. In our context, a false positive implies that a cat that would not have been adopted was classified as such, while a false negative implies that a cat that would have been adopted anyway was classified as one that would not have been.

It would be assumed that the cats that are at higher risk of not being adopted would receive more attention. Therefore, false positives here have a greater impact than false negatives. This is why it is better to optimize for specificity and precision rather than recall.

4. CONCLUSION

Apart from the variety of soft skills that we learnt by working together, like communication skills, management, and collaboration, the relevant learning outcomes have to do with the implementation of the solution.

Generally speaking, the specific techniques we used to improve the accuracy of our ensemble method like feature engineering and hyperparameter tuning resulted in small improvements in accuracy. The question that needs to be answered is was this small 2-4 % improvement worth the computational costs?

Using PCA to improve feature selection does improve the performance, but the efficacy of using it especially with regards to categorical variables still remains to be seen. We have doubts about whether it was useful in the long run.

Moving on to hyperparameter tuning, specifically using GridSearch. GridSearch was highly time consuming to use, and thus, only a few parameters were defined in the range. This brings the value proposition into question because the improvements in accuracy was only marginal.

Then again, we didn't know in advance whether these techniques would result in massive improvements in accuracy, so it was more of a learning experience regarding what works and what doesn't.

As far as future improvements are concerned, some potential avenues of consideration are:

- Using High Correlation Filter instead of Principal Component Analysis for feature selection
- Using RandomSearch instead of GridSearch over a wider selection of parameters to reduce the cost with regards to time
- Of course, we could always try another combination of machine learning models and develop a hybrid that could better solve this problem

TASK DISTRIBUTION

TANVEER MAHMOOD HASAN	1725413	LITERATURE REVIEW, VIDEO PRODUCTION
MUHAMMAD HAZIQ ADLI BIN ZAMZURI	1814981	LITERATURE REVIEW, PRESENTATION MGMT
MOHAMMED RAASHID SALIH BATHA	1817209	REPORT, DIRECTION AND INTEGRATION
FADHLUDDIN BIN SAHLAN	1817445	IMPLEMENTATION, METHODOLOGY
YASIN MOHAMMED	1814111	IMPLEMENTATION, METHODOLOGY

5. REFERENCES

- Koehrsen, W. (2018, January 10). Hyperparameter Tuning the Random Forest in Python. Retrieved January 01, 2021, from https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74
- Chakure, A. (2020, November 6). Decision Tree Classification - The Startup. Medium. https://medium.com/swlh/decision-tree-classification-de64fc4d5aac
- Kho, J. (2019, March 12). Why Random Forest is My Favorite Machine Learning Model. Medium. https://towardsdatascience.com/why-random-forest-is-my-favorite-machine-learning-model-b97651fa3706#:%7E:text=Random%20forests%20is%20great%20with,working%20with%20subsets%20of%20data.&text=It%20is%20faster%20to%20train,work%20with%20hundreds%20of%20features