BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

PetFinder Adoption Prediction

– ITSG report –

Team members
Bumbar Ioana
Bumbar Samuel
Dinga Madalina
Popoviciu Brigitta
Software Engineering
Group 258/1

2019

Abstract

This paper presents an intelligent system designed to aid pet adoption by providing information about certain animals to caretakers and also to people looking for a pet. One can search through all the available pets in the system and find out the best fit for him, based on its characteristics and also on its adoption chances, which the system can predict using machine learning algorithms (tree-based algorithms, multilayer perceptron, convolutional neural network). Specifically, the system can find the least "popular" animals, raise awareness of people and help caretakers focus on the animals that need most their help.

The first chapter of this paper focuses on obtaining a reliable algorithm that can predict the "adoptability" of an animal based on its characteristics, description and pictures. The scientific problem was approached by using supervised machine learning. Training the algorithms required a robust, labeled dataset, provided by PetFinder.my, Malaysia's leading animal welfare platform since 2008, with a database of more than 150,000 animals - `https://petfinder.my`.

The second chapter shows the related work that was taken in this direction and the results obtained by others, most of them shared by the participants at an online competition held on the Kaggle platform, called PetFinder.my Adoption Prediction [1]. It also presents useful tools in solving the given prediction problem.

The third chapter focuses on several approaches, which were implemented and compared during this research, for the final solution to be chosen and then integrated into a web application.

The fourth and final chapter presents a web application which uses the best solution found during this research, together with its functionalities.

Contents

Chapter 1

Introduction

1.1 Problem Overview

Thousands of homeless animals in the world need a family to adopt them. While many of them live in shelters all their life, others are euthanized and only a few of them end up finding a family willing to adopt them. However, in recent years, adoption has increased and many people have found a pet friend inside shelters, offering it a second chance. The aim of this work is to further increase the adoption rate by proposing an intelligent instrument that can work with centralized data of animals living in shelters, do a smart analysis of their real chance of adoption and provide useful information to the caretakers and also to people looking for a pet. In other words, the adoption system uses an intelligent approach when examining the animals, by learning to predict the chances of a pet to be adopted, in order to improve the work of animal shelters.

1.2 Paper structure and original contributions

The main focus in this paper is finding a viable solution based on machine learning algorithms that can predict the adoption speed for an animal with a high degree of accuracy. Another important step is integrating the final solution into a user-friendly application that caretakers and also people willing to adopt an animal can easily access via the internet.

The paper is structured in chapters as follows:

- The first chapter focuses on obtaining a reliable algorithm that can predict the "adoptability" of an animal based on its characteristics, description and pictures, by using supervised machine learning.

- The second chapter talks about related work, results and useful tools in solving the research problem.
- The third chapter focuses on several approaches, which were implemented and compared during this research and on the final solution which was then integrated into a web application.
- The fourth chapter presents the intelligent system as a whole, focusing on the integration part and the client, a web application which uses the best algorithm which can predict the "adoptability" of an animal. The functionalities are presented here, together with the usecases aimed to both caretakers and people looking for a pet.

Besides, in terms of contributions to the scientific problem, the main point is the implementation of an intelligent algorithm for solving the problem of predicting the adoption rate of rescued animals (i.e., the adoption speed - a value between 0 and 4). Specifically, supervised machine learning is used on a large dataset containing information about pets, which was collected by PetFinder.my, Malaysia's leading animal welfare platform since 2008, with a database of more than 150.000 animals - https://petfinder.my. The experiments reported in this paper were conducted on different machine learning models, such as convolutional neural networks, multi-layer perceptrons and tree-based algorithms, using different architectures and tuning the models in order to find the best algorithms. Also, combining the power of these models was taken into consideration in order to obtain predictions as accurate as possible.

Moreover, the second contribution of this report consists in building an intuitive, easy-to-use and user-friendly software application.

Chapter 2

Related work and useful tools

2.1 Related work

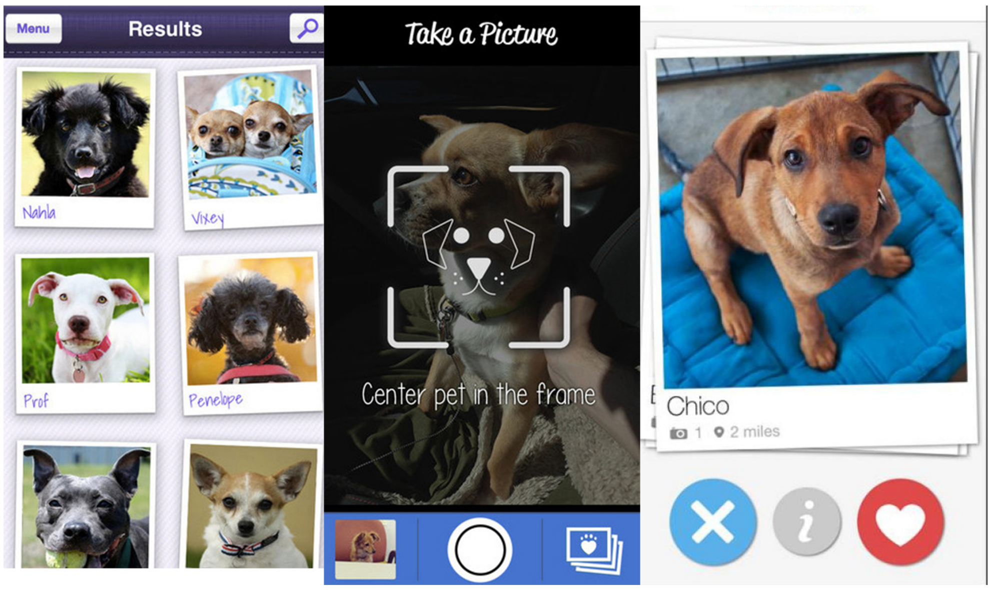
Thanks to the fact that the number of pet lovers are constantly increasing, the society fashion and city administrations as well provide more opportunities for us to have a comfortable life with our beloved pets. As for the chances of people adopting and keeping pets in their life is increasing as well. Therefore, the internet adapting to the newest trends, now has incomparably more information and together with that, many applications specialized in this field.

As of the purpose of our platform, we support the adoption over continuous breeding as long as there are orphan pets waiting for a loving parent. Considering that our initiative is not new, nor shocking to exits, we researched the domain a bit and found several applications, more or less complex, but with similar target as ours: adopt the perfect furry new family member.

For a better overview, we list the most closely related applications found, summarize and compare the functionalities with our vision.

First in the list, **Petfinder** is one of the applications that stand out more, due to their provided data. Owners of not only a website, but nowadays a mobile application as well, their main methodology is filter based search with a customer friendly interface. Main reason of our highlight, is their founded research of data mining projects, which provided us and many others with our current database of pets and their properties. Therefore, our goal and their goal is aligned, by creating an AI, based on their data, and find out as accurately as possible which pet is more likely to be adopted.

Their problem, as I already specified earlier, is listing the possible pet candidates, based



on different filters, such as location, breed, and others. Our solution wants to improve similar concepts further, by using our learning patterns and therefore showing the most likely adoptable pets, and drawing attention toward pets with a low scored profile as well.

Secondly placed, **LikeThat Pets**, is the application which when it comes to good ideas of how to use up artificial intelligence best, takes the lead. *The application allows you to take a photo of a dog or cat, and with the help of recognition technology, it will generate similar pets for you which are up for adoption* [1] . Almost like a facial recognition application, built for pets, it is a filtering method raised to a completely different level. Although, our problem might look highly different, both looks for most likely adoptable features, one at a user level, while our at a general level. While LikeThat Pets filtering can be considered very strong and discriminating, our method will give more chances to any pet up for adoption, balancing the adoptability percentages with listing both highly ranked pets and lowly ranked pets head to head, raiding awareness of no discrimination between any pet in need, but also taking in account user needs and preferences.

And last but not least, **BarkBuddy**, the tinder for pets and owners looking for a pet. Although no smart learning is present in the background, what our solution focuses more one, their preference based application can be a very good start for a future improvement towards pet adaptability. The left-right swiping when it comes to a pet you prefer or not, gathers good data about the properties of pets which are more "liked". The issues and problem they have is similar to, the earlier mentioned, Petfinder application, but with a different interface. Our application and idea, can be considered an impovement in the same ways, it can be considered for Petfinder app.

2.2 Useful tools

In order to build, deploy the models and build the web application, several libraries, IDEs and other tools were used:

- Jupyter Notebook - as data science environment
- scikit-learn - Python machine learning library [2]
- Tensorflow - machine learning library [3]
- Keras - Python neural-network library [4]

Chapter 3

Scientific Problem

3.1 Problem definition

In the current case, the scientific problem which needs to be studied and solved before building the software system that could improve the speed of adoption in animal shelters concerns the prediction of the adoption rate of rescued animals. Supervised machine learning can be used in this case, using the labeled dataset provided by the PetFinder.my platform mentioned earlier and published on the Kaggle AI platform [1].

3.2 Input data

The dataset contains two subsets - train and test data, presenting information about 14993 and 3948 pets respectively. Data about each pet is stored in different formats: tabular files, JSON format (for image metadata and sentiment analysis) and image data.

The dataset consists of:

- basic features of different pets - categorical (nominal, ordinal) and numeric (age, quantity, fee, video and photos amount).
- sentiment metadata extracted from the available descriptions for each pet using the Google Natural Language API [5]. It evaluates characteristics such as the power of sentiment for a certain description, using the score (i.e., the emotional leaning of the text - a normalized value between -1 and 1) and the magnitude (the overall strength of emotion).

- image data - extremely diverse, present in two forms: multiple raw images for each pet, available in different sizes and image metadata, extracted using the Google Vision API [6]. The metadata includes dominant colors and details about objects recognized in the images.

3.3 Output data

The problem of predicting "adoptability" can be reduced to a value between 0 and 4, representing the categorical speed of adoption for a pet. In this case, lower is fastest, meaning that animals having this value 0 have the highest chances of being adopted.

Chapter 4

Proposed approach

Describe your approach!Describe in reasonable detail the algorithm you are using to address this problem. A psuedocode description of the algorithm you are using is frequently useful. Trace through a concrete example, showing how your algorithm processes this example. The example should be complex enough to illustrate all of the important aspects of the problem but simple enough to be easily understood. If possible, an intuitively meaningful example is better than one with meaningless symbols.

Explain the experimental methodology and the numerical results obtained with your approach and the state of art approach(es). Try to perform a comparison of several approaches. Statistical validation of the results.

4.1 Data pre-processing

Like any real-world data, the dataset we chose to work with was unstructured, massive and likely to contain errors. In order to assure high quality of the data, pre-processing steps were compulsory in cleaning and transforming the data.

4.1.1 Strategies and techniques

The techniques that were used in the pre-processing stage are the following:

- **Data integration:** Data from various resources (i.e., basic features, sentiment data, image metadata) was combined. In this process, a lot of attention was given to object matching and redundancy. As a result, the final input that was fed to machine learning algorithms was contained inside one large CSV file containg 60 features.

- **Splitting up the data:** The dataset was split into training (80%) and testing data (20%) to be able to evaluate how the models perform with unseen data.
- **Normalization:** Normalization of numerical attributes was performed using the scikit-learn library in order to make sure that every attribute has an equal impact in the prediction.
- **Text mining:** Splitting up the description feature enabled the creation of new features, which were fed to the Google Natural Language API [5].
- **Aggregation:** Applying statistical means, sum to existing features contained in the sentiment dataset to obtain new features (e.g., doc_score_Mean, sen1_score_Mean, sum_mag_Mean).
- **Attribute subset selection:** The dataset was reduced by removing certain features (e.g., sentiment related features) in order to test how different models perform on a smaller subsets of the data.

4.1.2 Image Pre-processing

Image pre-processing included mainly image resizing before feeding the images to learning algorithms, such as convolutional neural networks. This step was necessary to reduce computation complexity and reduce space.

4.1.3 Feature Engineering and Feature Importance

Moreover, before feeding the data to a model, an essential step was done, namely feature engineering. Large majority of models performed better when using the additional extracted data. An essential contribution to new features came from the sentiment and image metadata. Regarding the image metadata, since some pets have more than one picture, the sum and the average of the scores were computed as new features, setting them to 0 for those who do not have pictures at all.

Additionally, feature importance has been analysed using algorithms such as random decision forest.

4.2 Models Built

Several models using different machine learning algorithms have been implemented using libraries such as scikit-learn [2], Tensorflow [3] and Keras [4], trained and tested, both locally and on Kaggle. Among the models tested, there were: Multilayer Perceptron Classifier, Convolutional Neural Networks (regression and classification approaches), Random Forest Classifier, K-Nearest-Neighbor, Support Vector Machine.

- **K-Nearest-Neighbor (KNN)**: is an effective and simple but not so much efficient sample-based lazy machine learning approach. The algorithm classifies objects into predefined majority classes of sample members that were created by machine learning. K-Nearest Neighbor searches for the most similar object from the samples with the Euclidean distance which calculates the distance of two vectors [2] [3].

  The benefit of this method is, that it does not require the use of training data to perform classification. The training data is kept in memory for later testing or validation. Also, it produces valid and valuable results compared to other algorithms with the right mix of prior knowledge. But there are some disadvantages, too. The complexity of its sample similarity computing is huge, so a lot of processing power is needed. In addition, the K-Nearest-Neighbor performance is easily affected by noisy samples and K-NN does not build the classification model itself [4].

  In order to create the model, we use from the **sklearn** library, the **KNeighborsClassifier** method, with a manually set 'k' value to 3. Future optimizations can be made here by *optimizing the 'k' parameter* between a range of one and one hundred to find the best 'k' value used.

  ```
  neigh = KNeighborsClassifier(n_neighbors=3)
  neigh.fit(train, adoptionSpeed_train)
  knc = neigh.predict(test)
  ```

```
}
```

- **Random forest:** is a supervised machine learning algorithm that is capable of both regression and classification tasks. The RF builds up multiple de-correlated decision trees using algorithms such as information gain and Gini index [1], rather than averaging the prediction of trees. There are two main concepts, first each tree in a random forest is trained from a random sample of data points, but some samples could be used multiple times in a single tree. That results in a lower variance but does not increase the bias (also known as bagging). Secondly, there is only a subgroup of all the features examined for splitting each node in the tree. After which the predictions of each individual tree is averaged to get the final outcomes [5] [6]

$$Gini = \sum_{i \neq j} p(i)p(j)$$

Figure 4.1: Gini index

The advantages of the Random forest are that it is very user-friendly, because it only has two parameters. One is the number of variables in the random subset at each node and the other is the number of trees in the forest. Moreover, RF can handle missing values and maintains accuracy for missing data. Furthermore, the method is resistant to noise or over-fitting, since re-sampling is not predicated on weighting. Also, it doesn't need as much processing power as methods based on only boosting and can handle large datasets with higher dimensionality. The disadvantages of Random forest are that like a black box approach, you have very little control on what the model does and it also could be slow to train [5] [6] [7]

In order to create the model, we use from the **sklearn** library, both the classifier and the regression models:

For the classifier **RandomForestClassifier** method has been used, with a manually set 'estimator' value to 500, and entropy criterion.

[1]**Gini index** The mean decrease in Gini coefficient is a measure of how each variable contributes to the homogeneity of the nodes and leaves in the resulting random forest.

```
best_forest = RandomForestClassifier(n_estimators=500, criterion='entropy')


best_forest.fit(train, adoptionSpeed_train)
forest_predicted = best_forest.predict(test)
```

For the regression, **RandomForestRegressor** method has been used, with a manually set 'estimator' value to 1000 for further testing purposes, and mse criterion.

```
rf = RandomForestRegressor(n_estimators = 1000, random_state = 42,
    criterion='mse')
rf.fit(train, adoptionSpeed_train);
forest_regr = rf.predict(test)
```

Future optimizations can be made on different estimator numbers optimizing it to find the maximal depth between minus one and one hundred of the Random forest and the number of trees in the number range from one to one hundred.

- **Support Vector Machine:** is a supervised machine learning algorithm that is capable of regression and classification tasks. The support vector machine builds a hyper-plane, which can be used for classification and is good for finding the boundary to separate different groups and maximize the margin. The goal is to choose the hyper-plane with the highest margin between the hyper-plane and the training set [8] [9].The benefit of the support vector machine algorithm is its effectiveness in high dimensional spaces and its ability to have different kernel functions for various decision functions. Furthermore, it has a good accuracy and works well on smaller cleaner datasets. A disadvantage of the support vector machine is, that it isn't working as well on large datasets due its high training time. It also has a lower effectivity on noisier datasets with overlay classes [10].

  In order to create the model, we use from the **sklearn** library and the **SVC** method.

```
clf = SVC(gamma='auto')
```

```
clf.fit(train, adoptionSpeed_train)
svc_predict = clf.predict(test)
```

To optimize the model further, parameter can be optimized to find the best values of 'gammaâ, because it is a crucial procedure to get the optimal parameters.

- **Multilayer Perceptron:** is inspired of the structure and functional aspects of the human brain (especially how brain neurons work) and uses this to develop algorithms that can model prediction problems and complex patterns.The algorithm is based on an input layer, multiple hidden layers and an output layer. Each layer consists of an interconnected group node, which is usually connected in a feed-forward way and afterwards is trained based on a back-propagation algorithm. The benefits of ANNs are the ability to model complex and nonlinear relationships, making them very useful on real life data. After a successful training, the algorithm will learn to generalize, so it can perform on unseen data. [11]

  Add code snippet and library description

  ```
  code snippet
  ```

- **Convolutional Neural Network (CNN):** Write description

4.3 Model evaluation

After the construction of a model based on training data, we started to evaluate the performance of our model on test data In the evaluation process, various performance measurements were used, such as accuracy, precision, recall and F1-score for classification and mean absolute error, R-squared for regression. The measurements are introduced in detail below:

(1) Accuracy and error rate: Models are evaluated by looking at correctly and incorrectly classified instances. Focus on the predictive capability of a model using confusion matrix.

(2) Precision p is the number of correctly classified positive examples divided by the total number of examples that are classified as positive.

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$Error\ rate = 1 - accuracy$$

$$p = \frac{TP}{TP + FP}$$

(3) Recall r is the number of correctly classified positive examples divided by the total number of actual positive examples in the test set.

$$r = \frac{TP}{TP + FN}$$

(4) F1-Score combines precision and recall into one measure

$$F1 = \frac{2pr}{p + r}$$

$$F1 = \frac{2}{\frac{1}{p} + \frac{1}{r}}$$

4.4 Results and Final Model

As a result, after several experiments using different subsets of the training set, tuning the parameters of the models and trying different neural network architectures, the best model was the convolutional neural network performing classification. Models were tested on both the full dataset and also using various subsets, containing the best 15/20 features. The difference in score was not significant. As a result, further work was made using the entire dataset.

The table below shows the results of the evaluation using the entire set of features for tree-based algorithms and multilayer perceptron, respectively the raw images as input for convolutional neural networks.

Considering the numbers obtained, **Figure 5.3** shows algorithms that performed well, as well as algorithms that were no better than random. Accuracy scores ranged from % to %, precision scores from % to % and recall scores from % to %. Unbalanced data could influence the accuracy of classification in this case, hence, without balancing data, the accuracy could be resultsofevaluation.png

Figure 4.2: Model Comparison

unrealistically higher. F1 score was used in the evaluation process. According to the accuracy and F1 score, random forest, SVM and ANN achieved the best results among all the classifiers, ?? HOW MUCH respectively //WRITE THE BEST ONCE . We could see that //THE BEST ONE is the most suitable method for our task and that with an accuracy of almost %. It is arguably a good method for predicting adoptability.

write about the final model

Chapter 5

Application

5.1 Functionalities

The application provides several functionalities, creating the chance for the people to find the right pet and also for the pets to find the right owner.

We will have the following functionalities:

- Registration / Authentication;
- List available pets;
- List most popular pets;
- List least adopted pets;
- Adopt a pet.

Registration / Authentication

One of the functionalities is the registration / authentication, which requires the basic details of the user in order to find to be able to proceed with the adoption of a pet.

List available pets

The app provides a list with non-adopted pets, which can be sorted by age, color, breed and some other criterias in order for the user to be more easily to choose the right pet.

List of most popular pets

The application also provides a list with the most popular pets, they being sorted by the adoptability score criteria, making the decision easily for the people who are not sure which animal to adopt.

List of least adopted pets

We also provided a list with the least adopted pets, which are the pets with the smallest adoptability score, helping the people who want to take a more interesting pet, which is not taken by everybody or for the people who feel that they want to help a pet with minimum chances to get into a family.

Adopt a pet

We have also a functionality that help the user to adopt the desired pet, specifying all the details that the user needs to know in order to proceed with the application for the adoption.

Chapter 6

Conclusion and future work

Try to emphasise the strengths and the weaknesses of your approach. What are the major shortcomings of your current method? For each shortcoming, propose additions or enhancements that would help overcome it. Briefly summarize the important results and conclusions presented in the paper.What are the most important points illustrated by your work? How will your results improve future research and applications in the area?

Bibliography

[1] Kaggle.com, “Petfinder.my adoption prediction | kaggle,” 2019, last accessed 09 March 2019. [Online]. Available: https://www.kaggle.com/c/petfinder-adoption-prediction

[2] Scikit-learn.org, “Documentation scikit-learn: machine learning in python â scikit-learn 0.20.3 documentation,” 2019, last accessed 10 March 2019. [Online]. Available: https://scikit-learn.org/stable/documentation.html

[3] TensorFlow, “Tensorflow guide,” 2019, last accessed 10 March 2019. [Online]. Available: https://www.tensorflow.org/guide

[4] Keras.io, “About keras models - keras documentation,” 2019, last accessed 10 March 2019. [Online]. Available: https://keras.io/models/about-keras-models/

[5] G. Cloud, “Natural language api,” 2019, last accessed 09 March 2019. [Online]. Available: https://cloud.google.com/natural-language/docs/basics

[6] ——, “The google vision api,” 2019, last accessed 09 March 2019. [Online]. Available: https://cloud.google.com/vision/docs/reference/rest/v1/images/annotate

Bibliography

[1] https://www.onegreenplanet.org/animalsandnature/apps-that-help-adoptable-animals-find-forever-homes

[2] Tan, Songbo.(2005). Neighbor-weighted K-nearest Neighbor for Unbalanced Textorpus. Expert Systems with Applications. 28. 667-671.

[3] Bruno, Trstenjak Sasa, Mikac Donko, Dzenana. (2013). KNN with TF-IDFbased framework for text categorization. Procedia Engineering.

[4] Li, Baoli Yu, Shiwen Lu, Qin. (2003). An Improved k-Nearest Neighbor Al-gorithm for Text Categorization.

[5] Liaw, Andy Wiener, Matthew. (2001). Classification and Regression by Ran-domForest.

[6] Breiman, L. Machine Learning (2001).Kluwer Academic Publishers.

[7] Cutler, David C Edwards, Thomas Beard, Karen Cutler, Adele T Hess,Kyle Gibson, Jacob Lawler, Joshua. (2007). Random Forests for Classificationin Ecology.

[8] Support vector machine simply explained.(2019, January 07).Retrievedfrom https://towardsdatascience.com/support-vector-machine-simply-explained-fee28eba5496.

[9] Support vector machines the linearly separable case.(2009, April 07). Re-trieved from https://nlp.stanford.edu/IR-book/html/htmledition/support-vector-machines-the-linearly-separable-case-1.html.

[10] Aylien,N. Support vector machines simple explanation. Retrieved fromhttps://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html.

[11] Mahanta,J.NeuralNetworkadvantages.(2017,July10).Retrievedfromhttps://towardsdatascience.com/in
to-neural-networks-advantages-and-applications-96851bd1a207

[12] Xin Yao, âEvolving artificial neural networks,â in Proceedings of the IEEE, vol.87, no. 9, pp. 1423-1447, Sept. 1999

[13] Neural Network operator. Retrieved from https://docs.rapidminer.com/latest/studio/operators/model
/neuranets/neuralnet.html