Udacity Machine Learning Engineer Nanodegree

Capstone Project

Marizu-Ibewiro Makozi

May 15, 2020

1. Definition

1.1 Project Overview

Machine learning is used in many fields today. In this project, machine learning will be used to build a model that can decide based on the role information of an employee whether that employee shall have access to a specific resource. An employee that uses a computer in order to do their tasks. needs access to certain areas of software programs or access rights to execute actions such as read, write or delete a document. While working, employees may encounter that they don't have a concrete access right required to perform the task at hand. In these situations, a supervisor or an administrator has to grant them access. The process of discovering that a certain access right is missing and removing that obstacle is both time-consuming and costly. A model that can predict which access rights are needed based on the current role of an employee is therefore relevant.

1.2 Problem Statement

The problem comes from the Amazon.com *Employee Access Challenge Kaggle Competition* [1] and is described as follows: *"The objective of this competition is to build a model, learned using historical data, that will determine an employee's access needs, such that manual access transactions (grants and revokes) are minimized as the employee's attributes change over time. The model will take an employee's role information and a resource code and will return whether or not access should be granted."*

This is a classification problem where the model takes a user's role information and a resource as input and produces the expected output whether the access to the resource will be granted or denied. This is also a supervised learning problem because the dataset is labelled. Solution:

1. Explore data in order to gain insights.

2. Split the data into a training- and testing set using a stratified split.

3. Train many different binary classification models using standard parameters.

4. Apply transformations or regularizations.

5. Compare plain models and transformed models.

6. Pick the three best models based on the performance metric.

7. Tweak the chosen models in order to improve model performance using K-fold cross-validation

8. Evaluate the tweaked models on the test set.

9. Conclusion

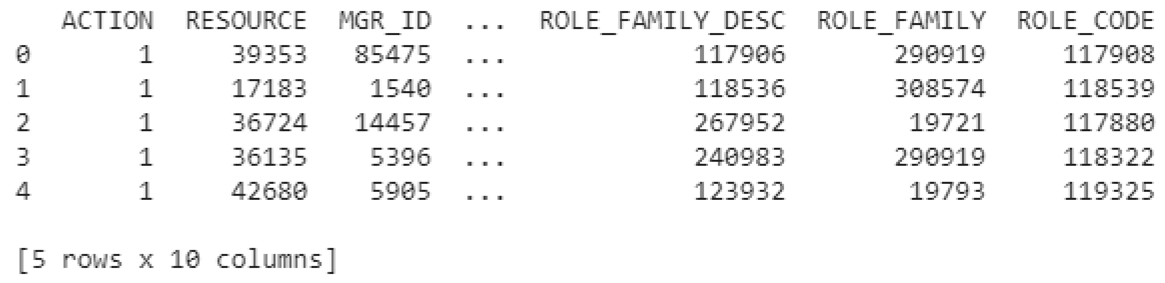
1.3 Metrics

To quantify model performance, the area under the ROC curve will be used. This metric is appropriate for this type of project because it works well even if the classes are not balanced. It was the metric of choice in the herein before mentioned Kaggle competition. The metric is derived by first constructing the ROC curve and then calculating the area under that curve. *"The ROC curve is created by plotting the true positive rate against the false positive rate at various threshold settings"* [2], where the threshold is a value between 0 and 1 that determines how sure the model needs to be in order to classify a data entry as positive (access granted in the problem at hand).

2 Analysis

2.1 Data Exploration

There are 32769 entries in the dataset with no missing values.



```
   ACTION  RESOURCE  MGR_ID  ...  ROLE_FAMILY_DESC  ROLE_FAMILY  ROLE_CODE
0       1     39353   85475  ...            117906       290919     117908
1       1     17183    1540  ...            118536       308574     118539
2       1     36724   14457  ...            267952        19721     117880
3       1     36135    5396  ...            240983       290919     118322
4       1     42680    5905  ...            123932        19793     119325

[5 rows x 10 columns]
```

Figure 1: shows the first five rows in the dataset.

The dataset has ten attributes. All attributes are categorial. One attribute called RESOURCE holds the ID of the resource for which the access has been granted or denied. There are 7518 different resources in the dataset. The target attribute is called ACTION. The other eight columns provide role information for an employee1:

• MGR_ID - ID of the manager of employee (4243 different values)

• ROLE_ROLLUP_1 - Role ID of employee (128 different values)

• ROLE_ROLLUP_2 - Second role ID of employee (177 different values)

• ROLE_DEPTNAME - Role department description (449 different values)

• ROLE_TITLE - Role business title (343 different values)

• ROLE_FAMILY - Role family description (67 different values)

• ROLE_FAMILY_DESC - Extended role family description (2358 different values) • ROLE CODE - Company role code; this code is unique to each role (343 different values)

2.2 Exploratory Visualization

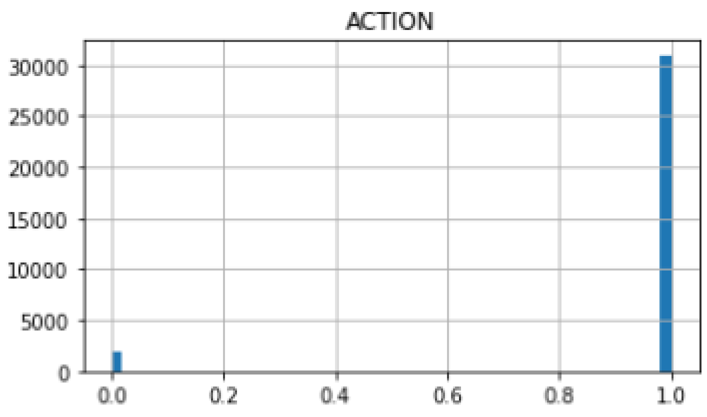


Figure 2: Histogram for target attribute

Figure 2 shows that the ACTION-attribute is highly unbalanced. More than 94% of the rows have an ACTION-attribute of 1 (access granted) whereas only roughly 6% have a 0 (access denied). The accuracy metric is therefore inadequate for this dataset because even a dumb model that always predicts 1 would have a very high accuracy.

2.3 Benchmark Model

An out of the box logistic regression model will be used as the benchmark for this project, because the model is fast, simple to implement and to interpret and should give far better results than random guessing for the problem at hand. Because the problems stems from a Kaggle competition, as a secondary benchmark, the result of the final solution will be compared to the result of the solution of the team that won the competition. The submissions to the competition were judged on the area under the ROC curve (AUC) metric. This metric will be used to compare the results. The winning team (Paul Duan & BS Man), who got an AUC value of 0.92360, which is an excellent result and the author will strive to come close to it.

3 Methodology

3.1 Data Pre-processing

Compared to the samples with a positive class (access granted) the dataset contains very few samples with a negative class (access denied). The first pre-processing step is therefore to put aside a test set that preserves the percentage of samples for each class. As mentioned before this is achieved by using a stratified shuffle split. If a standard random split had been used to split the dataset into a training and testing set, it would be possible that for example the training set would not contain a single negative sample which would have certainly a negative impact on the performance of the classifiers. The second pre-processing step is to one-hot encode the predictive attributes. This transforms all the categorial attributes to binary attributes and fixes the issue that the models would assume that two nearby values of a categorial attribute are more similar than two distant values.

3.2 Implementation

The programming language that was used to implement the solution for the project was Python[3] . The development Python code was written in a Jupyter Notebook[4]. Additional Python libraries and packages that were used during the development stage comprised NumPy[5], pandas[6] , SciPy[7] and Scikit learn[8]. For the XGBoost classification model the XGBoost12 Python package had to be installed. The first step was to make some imports and to load the dataset into a pandas dataframe by using the read csv method. A random state was set to a fixed value because this allowed to replicate the results when the code was executed multiple times.

```
# Imports
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, StratifiedShuffle
Split

# load data
random_state = 42
df = pd.read_csv('https://raw.githubusercontent.com/makozi/Udacity-ML-
Engineer-Capstone-Project/master/Data/train.csv')
```

Next is to create two distinct datasets for training and testing. The ratio between the positive and negative classes had to be preserved. The initial approach of simply using a random split was neglected. Instead a stratified split was executed. The size of the test set was chosen to be 25% of the whole data set. This left a fair amount of data for the training set and enough data in the test set to do a meaningful evaluation.

4 Results

4.1 Model Evaluation and Validation

To evaluate the model the previously set aside test-set was used. The parameters which had different values than the default value of the XGBoost model, that was found to be the strongest among those tested.

5 Conclusion

5.1 Reflection

The overall process can be summarized as follows:

• Analysis of the problem and data

• Splitting data into distinct training- and testing-sets

• Pre-process the data, i.e. one-hot-encode the categorial features

• Training different models on the training-set

• Picking the three models with the best performance on the training set

• Try to improve the best models by modifying the hyperparameters of the best three models

• Apply the pre-processing step to the test-set

• Evaluate the performance of the models on the test-set

• Compare the final models with the benchmark

One of the biggest problems or challenges of this project was that training and evaluating the models took a lot of time. Especially the SVM model is very computational expensive.

References

[1] Amazon.com – Employee Access Challenge Predict an employee's access needs, given his/her job role. https://www.kaggle.com/c/amazon-employee-access-challenge.

[2] Receiver operating characteristic https://en.wikipedia.org/wiki/Receiver operating characteristic

[3] https://www.python.org/

[4] https://ipython.org/notebook.html

[5] http://www.numpy.org/

[6] https://pandas.pydata.org/

[7] https://www.scipy.org/

[8] https://scikit-learn.org/stable/index.html

[9] https://xgboost.readthedocs.io/en/latest/