Udacity Machine Learning Nanodegree
Capstone Project

Carsten Krüger

June 17, 2018

1 Definition

1.1 Project Overview

Machine learning is used in a wide variety of fields today. Luca Talenti et al. [1] for example used a classification model to predict the severity criteria in imported malaria. 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 has to use a computer in order to fulfill 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 those 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 stems from the *Amazon.com Employee Access Challenge Kaggle Competition* [2] and is there 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 supervised learning problem because the dataset is labeled. Anticipated solution:

1. Explore data in order to gain insights.
2. Train many different binary classification models using standard parameters.
3. Apply transformations or regularizations.
4. Compare plain models and transformed models.
5. Pick the three best models based on the performance metric.
6. Tweak the chosen models in order to improve model performance.
7. Evaluate the tweaked models on the test set.
8. 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. Moreover 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" [3],

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). For example if the threshold was 0.7 the model would have to have calculated a probability of at least 70 % to classify a data entry as positive.

2 Analysis

2.1 Data Exploration

There are 32769 entries in the dataset with no missing values. Figure 1 shows the first five rows in the dataset.

	ACTION	RESOURCE	MGR_ID	ROLE_ROLLUP_1	ROLE_ROLLUP_2	ROLE_DEPTNAME	\
0	1	39353	85475	117961	118300	123472	
1	1	17183	1540	117961	118343	123125	
2	1	36724	14457	118219	118220	117884	
3	1	36135	5396	117961	118343	119993	
4	1	42680	5905	117929	117930	119569	

	ROLE_TITLE	ROLE_FAMILY_DESC	ROLE_FAMILY	ROLE_CODE
0	117905	117906	290919	117908
1	118536	118536	308574	118539
2	117879	267952	19721	117880
3	118321	240983	290919	118322
4	119323	123932	19793	119325

Figure 1: Top 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 employee[1]:

- `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)

[1] `https://www.kaggle.com/c/amazon-employee-access-challenge/data`

Because the eight attributes that describe the role of an employee and the resource attribute are categorial, they will be vectorized. This is achieved by using one-hot encoding. For example the `ROLE_FAMILY`-attribute has 67 different categories. Each row in the dataset that has been one-hot encoded will contain one entry for each category. The entry will be 1 (hot) only for the entry corresponding to the current role family and 0 (cold) for the 66 other role families. This example shows that the number of input attributes will increase tremendously.

2.2 Exploratory Visualization

Figure 2 shows that the `ACTION`-attribute is highly unbalanced. In fact 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.

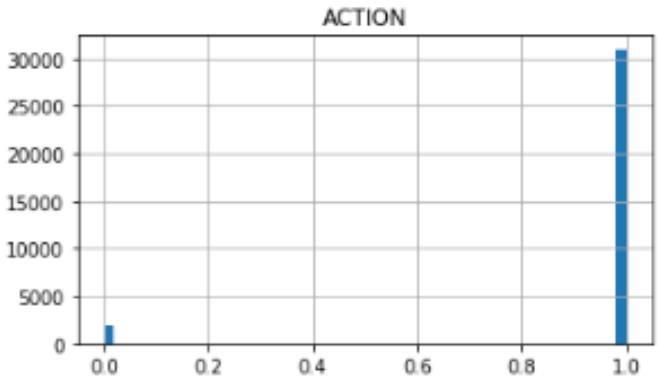

Figure 2: Histogram for target attribute

2.3 Algorithms and Techniques

First of all it is important to put aside a test set. This set will not be touched until the very end when all models have been trained and optimized. To generate the test set stratified shuffle split[2] will be used which preserves the percentage of samples for each class. A preprocessing pipeline will be created, that selects role attributes and applies one-hot encoding to them. This step is important because, as mentioned before, the attributes are categorial and two values that are close to each other are not more similar than two values with a larger distance. Different binary classification models with standard parameters will be created. The classifiers trained in this project are Logistic Regression, Decision Tree, SVM, Random Forest, AdaBoost and XGBoost. They will be initialized with a fixed random seed in order to make the results reproducable. There performance will be measured using scikit learn's cross validation score[3] function in order to circumvent overfitting. The best models will be tweaked by trying different sets of parameters. The goal is to furhter increase their performance. The best set of hyperparameters for the different models will be determined by doing a grid search[4]. The grid search function will try out all different combinations of hyperparameters and values that it gets passed as an argument using cross-validation. This is far more convenient than the tedious task of trying out a bunch of hyperparameters manually. Finally the previously generated test set will be used to determine whether the final model generalizes well.

2.4 Benchmark

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

[2] `http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html`
[3] `http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html`
[4] `http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`

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. Therefore this metric will be used to compare the results. The winning team got an *auc* value of 0.92360, which is an excellent result.

3 Methodology

3.1 Data Preprocessing

Compared to the samples with a positive class (access granted) the dataset contains very few samples with a negative class (access denied). The first preprocessing 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 preprocessing step is to one-hot encode the predictive attributes. This is done by using sklearn's *OneHotEncoder*[5]. 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 first step is to make some imports and to read the dataset:

```
import numpy as np
import pandas as pd
random_state = 42
df = pd.read_csv('../data/train.csv')
```

The second step is to create the stratified training and test sets:

```
from sklearn.model_selection import StratifiedShuffleSplit
sss = StratifiedShuffleSplit(n_splits=1, test_size=0.25, random_state=random_state)
for train_index, test_index in sss.split(df, df['ACTION']):
    train_set, test_set = df.loc[train_index], df.loc[test_index]
```

The third step is to extract the labels and the attributes for the training set:

```
access = train_set.drop('ACTION', axis=1)
access_labels = train_set['ACTION'].copy()
```

The fourth step uses a `DataFrameSelector`:

```
# DataFrameSelector class, taken from "Hands-On Machine Learning with Scikit-Learn &
# Tensorflow by Aurlion Gron (O'Reilly). Copyright 2017 Aurlion Gron, 978-1-491-96229-9,
    Page 41
from sklearn.base import BaseEstimator, TransformerMixin
class DataFrameSelector(BaseEstimator, TransformerMixin):
    def __init__(self, attribute_names):
        self.attribute_names = attribute_names
    def fit(self, X, y=None):
        return self
    def transform(self, X):
        return X[self.attribute_names].values
```

[5] http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

With this class a pipeline is created that one-hot encodes the attributes of the training set:

```
# get attributes
attributes = access.columns.values.tolist()

from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder

# one-hot encode the categorial attributes
pipeline = Pipeline([
    ('selector', DataFrameSelector(attributes)),
    ('encoder', OneHotEncoder())
])
access_1hot = pipeline.fit_transform(access)
```

3.3 Refinement

4 Results

4.1 Model Evaluation and Validation

4.2 Justification

5 Conclusion

5.1 Free-Form Visualization

5.2 Reflection

5.3 Improvement

References

[1] L1 logistic regression as a feature selection step for training stable classification trees for the prediction of severity criteria in imported malaria

Luca Talenti, Margaux Luck, Anastasia Yartseva, Nicolas Argy, Sandrine Houz, Cecilia Damon

arXiv:1511.06663 [cs.LG]

[2] 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,

[3] Receiver operating characteristic

https://en.wikipedia.org/wiki/Receiver_operating_characteristic