SAKI SS 2021 Homework 1

Author: Tim Löhr

Program code: https://github.com/Mavengence/SAKI_Homework_1

Summary

Background

For my university master course Software Applications with Artificial Intelligence, I received a dataset of bank transfer data with 209 datapoints and 11 features. The features are containing basic transactional information, such as the amount of money, the banking number or usage purpose. The data is provided by Adorsys GmbH.

Objective

This project aims to quantify how accurately bank transfer data can classify seven different labels. Those labels are for example standard of living or income. Different data from the bank transfer could have an indication for which label this payment aims for. Therefore, I analyses all the features given to us and underdo different preprocessing techniques and combinations, because we have categorical data, numbers and also text.

I try out different combinations with the data and use the naive bayes classifier for the classification task.

Furthermore, the dataset is split into a train, test and validation set, because of the slight data imbalance.

In the end, the trained model will be evaluated based on the accuracy and F1 score, because we have a multiclass classification task which is imbalanced. Therefore, only the accuracy score is not sufficient enough to evaluate the result.

Data Preprocessing

The data has the shape of 209 rows and 11 features. One row has mostly nan values for the most important features I am using, therefore I drop this row. Other than that, there exist 40 rows where the feature Auftragskonto is missing, but since I am using four features in total, I still kept those rows, because the information gain was still bigger than dropping 20% of the dataset.

Training Features

For this project, I tried to understand the features and combining them to the best possible outcome. It appeared, that the combination of Buchungstext, Kontonummer, Betrag and Bankleitzahl works the best for me.

Buchungstext has 19 unique values, where 13 of them uniquely are used for one specific label. This means, that only six of the nineteen features classify multiple labels. This allows for precise predictions.

Kontonummer has 53 unique values, where 53 of them are uniquely used for one specific label. Same as for the Buchungstext, this is a very useful feature to train on and therefore after my analysis I included it into the feature set.

Bankleitzeit has 44 unique values, where 40 of them uniquely are used for one specific label. Same as for the Buchungstext and Kontonummer, this is a very useful feature to train on.

Plus Minus Betrag is an artificially created feature in which I am using the sign of the feature Betrag. At first, I tried to use Betrag, but because for obvious reason, it makes no sense to use Betrag, which is actually a numerical feature, as a categorical feature. Therefore I binned the numerical values into [-5000, -2500, -500, -50, 0, 50, 500, 2500, 5000] these nine bins. It appeared that it decreases the accuracy, but only the sign of plus or minus is increasing the accuracy.

Other Features

I tried out most of the different possible features. I used for example a NLP pipeline to make use of the Verwendungszweck, but it only decreased the accuracy. The Valueatdatum also did not offer any increase in accuracy either, therefore I dropped it to keep the features as few as possible. Also Beguenstigter/Zahlungspflichtiger did not improve the accuracy at all. For those text features, a more in-depth NLP needs to be done. I therefore used only categorical data that can easily be one-hot-encoded.

Model Training

The model is the gaussian naïve bayes classifier provided by sklearn. This classifier offers one parameter to optimize on, the variable smoothing. Due to the dataset being very small (208 rows), I am using a repeated stratified K-Fold algorithm in combination with a GridSearch for the optimization of the variable smoothing and to be sure, that the accuracy I achieve is stable and not due to a lucky shot very high. The stratified K-Fold algorithm is important, because of the imbalance of the dataset. I am using four splits with five repeats. This method gives a stable outcome of the accuracy of the classifier, because the labels are distributed the same for the four splits.

Evaluation

Train, Test and Validation Sets

For the evaluation, I created a trainset, validationset and a testset. Because the dataset is slightly imbalanced towards the leisure and standard of living label, I created the testset in such a way, that it is completely balance. It consists of five rows per label, so 30 rows in total. The validation set has the same balance as the train set.

Accuracy

```
Train Accuracy: 1.0
Validation Accuracy: 0.94
Test Accuracy: 0.67
```

This result shows, that the data is overfitted towards the imbalanced data, because the balanced testset is not able to perform good. Even though we have six labels, an accuracy of 67% shows somewhat of a “training”, it is not nearly as good as the 94% of the validation set. Also, the 100% accuracy of the trainset suggest that the model is heavily overfitted. This is most likely due to the labels, because as I explained earlier, the features are not often overlapping over the features.

F1-Score

It is very interesting to see, that the imbalanced features private and standard of living perform the worst on both precision and recall. This makes total sense, due to the fact that the data is overfitted towards the imbalance. The F1-Score for both features is bad, because we have many false positives. This makes intuitive sense, because the model is trained to predict in an imbalanced way towards private and standard of living.

Result

My project showed, that I did the mistake of disregarding the imbalance of the trainset. The validation accuracy performs with 94% rather high, but the evaluation shows, that this model is not ready to be used in the real world. I avoided the rebalancing in the first place, because if I used a balanced trainset, I either need to oversample or undersample. Oversampling is creating artificial features, especially for our case this generates heavily wrong results. Because we already have so few data, I did not want to undersample either. I assume, that if proper NLP is used, the accuracy can be heavily increased by creating new features with the text. My test accuracy of 67% shows, that some kind of learning took place.

Literature

https://scikit-learn.org/stable/

https://www.scikit-yb.org/en/latest/api/classifier/classification_report.html

https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da

Screenshot

3.1 Train, Test and Validation Accuracy

```
print(f"Train Accuracy: {accuracy_score(gs_nb.predict(X_train), y_train)}")
print(f"Validation Accuracy: {np.round(accuracy_score(gs_nb.predict(X_val), y_val), 2)}")
print(f"Test Accuracy: {np.round(accuracy_score(gs_nb.predict(X_test), y_test), 2)}")
```

```
Train Accuracy: 1.0
Validation Accuracy: 0.94
Test Accuracy: 0.67
```

```
plt.hist(gs_nb.predict(X_test))
plt.title("Test Prediction Balance")
plt.xticks([0, 1, 2, 3, 4, 5], classes)
plt.show()
```