Report

December 6, 2020

1 Classification project work report

1.1 Introduction

The goal of the project is to build a classifier that predicts the class (event type) of the NPF data's observations. This is achieved by first analyzing the provided data. After that, by selecting an appropriate machine learning model and possibly a dimensionality reduction algorithm. Training the model with the provided labeled data, and then testing it by predicting the also given, unlabeled, NPF data.

The main part of the goal is to create a well-working binary classifier. However, good scoring in multinomial classification is a secondary goal. In our project, we held these two flags equally high during feature selection and while choosing the ML model. For this reason, we ended up with one particular classifier for both cases out of 12 different ones.

In this report, we will first go through the process of completing the project work. So first, we will discuss how we analyzed the NPF data. After that, we will explain what machine learning pipeline we ended up choosing and how the selection process proceeded. We will also talk about the optimization of hyperparameters of our model and the dimensionality reduction technique. Lastly, we will have a look at the results of our classifier and discuss the project as a whole.

1.2 Data analysis

In this chapter, we will explain what steps we took at the begging of the project to analyze and understand the data at hand. Originally we only knew that we were dealing with both labeled and unlabelled NPF data. From which latter one, we were supposed to predict using the first one as training data for supervised machine learning.

The first thing we did after loading the data was, of course, to have a simple plain look at what we are dealing with. Because of this, we then knew that we had data that had columns for means and standard deviations of multiple variables. The labeled data also contains the class (event type) and date of the observation, unlike unlabelled data, which doesn't contain either of them because they are pruned away. We did notice that the mean and std columns of each variable clearly have different distributions regarding the values they contain.

After that, we proceeded by visualizing the data using different plots. One of the more insightful ones was a scatter plot representing the correlation between variables. We executed this for most of the data set and found out that means and stds of variables that are similar to each other have a high correlation between each other. However, this correlation doesn't expand between stds and means nor between completely different variables. So basically, the data set contains groups of columns that highly correlate with each other.

However, to be sure about the correlation between columns in the data set, we computed the correlation matrix of the data sets. Results did support the observation done using plots.

1.3 Selecting ML model and dimensionality reduction technique

Analyzing the data didn't give out much regarding how it correlates with the target of the classification (event type). For that reason, we took the more computationally harder route to achieve the best results in classifying the NPF data. Because we had to go through many different classifiers, we also decided to combine them all with a bunch of different dimensionality reduction techniques.

For evaluating the classifiers and their performance with different dimensionality reduction techniques, we introduced two parameters. First is the accuracy score of the model using cross-validation, and second is how well they perform in both multinomial and binary classification tasks. To compare the usefulness of dimensionality reduction techniques, we performed cross-validation for all of the models with full data set too. However, the parameters of the dimensionality reduction techniques were optimized before the actual model selection began. However, we will talk about that in the next chapter of the report.

We computed the accuracy score of cross-validation as a mean of n independent repetitions of k-fold cross-validation using labeled data. Because of the limitations of computers, for model selection, we used three and ten for n and k, respectively.

The results of plenty of computing were quite surprising. This is because the top 3 classifiers ended up being logistic regression, Gaussian process classifier, and Multi-layer Perceptron classifier wIth a variety of dimensionality reduction techniques. From them, the most consistently performing model was the Gaussian process classifier using the original data, so without any dimensionality reduction technique.

Even though it wasn't the most accurate one is neither binary nor multinomial classification tasks. In the binary case, the most well working model was a combination of gaussian process classifier and univariate feature selection using F-test estimate of the degree of linear dependency between two random variables. Whereas in the multinomial case, the most accurate model-technique combination was a pair of logistic regression and extra-trees classifier. From which, the extra-trees classifier was used to select useful features for actual classification.

In the end, because of all the evidence, we decided to choose the Gaussian process classifier as our ML model for NPF data. We did, however, run one more cross-validation just for the selected model but with n=10. The result of this experiment did support our initial results. The accuracy of the Gaussian process classifier in the binary classification task was 89.9% with a standard deviation of 0.043, and in the multinomial classification task, it was 67.5% with a standard deviation of 0.061.

1.4 Optimization of hyperparameters

Before we began searching for a proper machine learning model for the NPF data by iterating through a multitude of them in a combination of different dimensionality reduction techniques, and of course without them too as a baseliner of the model accuracy. We did optimize the parameters of the techniques by comparing their results on different parameters in both binary and multinomial classification tasks. The evaluation was conducted by performing the earlier mentioned cross-validation process using the logistic regression model as the classifier for the techniques.

After we had found seemingly optimal parameters for each dimensionality reduction technique in both cases, we did one more check-up iteration for them with higher n to make sure results didn't change. Because the last round of computation only strengthened our view on the parameters optimality, we decided to choose them for the actual model search.

1.5 Results

TODO: work in progress!

1.6 Conclusion

TODO: work in progress!

1.7 Appendix

1.7.1 Code

[1]:
```
import numpy as np
import pandas as pd

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF

from sklearn.decomposition import PCA

from sklearn.linear_model import LogisticRegression

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_classif
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import ExtraTreesClassifier
```

[2]:
```
df = pd.read_csv('npf_train.csv', index_col='date')
df.drop(['id', 'partlybad'], axis=1, inplace=True)

class2 = df['class4'].copy()
class2[class2 != 'nonevent'] = 'event'
df['class2'] = class2

df['class4'], mapping_class4 = df['class4'].astype('category').factorize()
df['class2'], mapping_class2 = df['class2'].astype('category').factorize()
```

[3]:
```
df_test = pd.read_csv('npf_test_hidden.csv', index_col='id')
df_test.drop(['date', 'partlybad', 'class4'], axis=1, inplace=True)
```

[4]:
```
k_fold_splits = 10
repeats = 5

def kFoldCrossValidation(model, data, labels):
    # evaluate model
    cv = RepeatedStratifiedKFold(n_splits=k_fold_splits, n_repeats=repeats,␣
 ↪random_state=1)
    n_scores = cross_val_score(model, data, labels, scoring='accuracy', cv=cv,␣
 ↪n_jobs=-1)

    print("mean of the accuracies: " + str(np.mean(n_scores)))
    print("std of the accuracies: " + str(np.std(n_scores)))
```

[5]:
```
X = df.drop(['class4', 'class2'], axis=1)

scaled = StandardScaler().fit_transform(pd.concat([X, df_test]))
X_train = pd.DataFrame(scaled[0:430], columns=X.columns)
X_test = pd.DataFrame(scaled[430:], columns=X.columns)

y_class2 = df['class2']
y_class4 = df['class4']
```

[6]:
```
kbest = ('kbestmutual', SelectKBest(mutual_info_classif, k=90))
selecttree = ('selecttree',␣
 ↪SelectFromModel(ExtraTreesClassifier(n_estimators=70)))

gaussian = ('model', GaussianProcessClassifier(1.0 * RBF(1.0)))
logistic = ('model', LogisticRegression())

binary = Pipeline([kbest, gaussian])
multi = Pipeline([selecttree, logistic])

main = Pipeline([gaussian])
```

[45]:
```
X_tr, X_te, y_tr, y_te = train_test_split(X_train, y_class4, test_size=0.33)
main.fit(X_tr, y_tr)
predicion = pd.Series(main.predict(X_te))
```

[46]:
```
guess = pd.Series(mapping_class4[predicion])
actual = pd.Series(mapping_class4[y_te])

guess = guess.astype(str)
actual = actual.astype(str)

guess[guess != 'nonevent'] = 'event'
actual[actual != 'nonevent'] = 'event'

np.sum(guess == actual) / guess.size
```

[46]:
```
0.8732394366197183
```

[34]:
```
main.fit(X_train, y_class4)

classes = pd.Series(main.predict(X_test))
p = pd.DataFrame(main.predict_proba(X_test))
```

[10]:
```
results = []

for i in range(0, classes.size):
    label = classes[i]
    p_value = p.loc[i, label]
    results.append([label, p_value])
```

[15]:
```
answers = pd.DataFrame(results, columns=['class4', 'p'])
answers['class4'] = mapping_class4[answers['class4']]

answers['class4'].value_counts()
```

[15]:
```
nonevent    566
II          254
Ib          127
Ia           18
Name: class4, dtype: int64
```

[12]:
```
answers.to_csv('answers.csv', index=False)
```

[ ]:
```

```