Machine Learning Engineer Nanodegree

Capstone Project

Malgorzata Kot
23.08.2018

I. Definition

Project Overview

Machine Learning can support diverse discplines and one of them is marketing. For example, according to research using machine learning models for customer classification results in higher conversion rates[1], better potential customer identification[2] or can be used for quick sentiment classification [3], which was previosuly done manually. One of the problems, marketing is facing every day, is predicting, who of the potential customers will buy a product. It helps in efficient allocation of ressources as well as getting to know the customer base. For companies with large numbers of customers with heterogenous characteristics, supervised machine learning can be used for predicting potential customer interest in buying a product.

In this project, potential customers will be classified based on bank marketing dataset from UCI Machine Learning Repository. (link) The goal is to predict the variable “y”, which explains if customer has subscribed a deposit or not, which is equivalent to: did he buy a product or not.

Problem Statement

I will classify potential customers of a bank based on an information if they bought a product or not. The classifier, after getting data from a potential customer, should return a 1 or 0, 1 meaning customer will buy the product and 0 he/she will not buy. Additionally, there will be probabilty returned of how likely it is, that customer will subscribe to the deposit.

Metrics

Main metric will be F-2 Score (derived from F-Beta score), but additionally it will be measured with accuracy and confusion matrix. Ideally all of them should be higher than in the base model.

Precision measures how many true prediction were made out all values predictied true (true positives and

false positives). Recall measures how many true positives were out of all true values in dataset. F-Beta score combines both precision and recall in order to find the balance between them. Because this is marketing data, where we look for potential opportunities, identifying all potential customers is more improtant than making sure, that they are no false identifications. That is why I will use F-2 score, where beta = 2, what give more credit to recall than precision.

Accuracy is the amount of correctly predicted values divided by all predictions and multiplied by 100. Accuracy is not the most useful measure for this problem. If the assumption in benchmark model is, that all contacted clients are willing to buy, the model would get a low accuracy (11%). But if the assumption would be, that none of called people would by a product, accuracy will be very high (89%). Where second predictions has a very high accuracy, it makes completely no sense, as we look for people who are willing to buy a product. First prediction makes more sense, but the number is very low and easy to improve. So if final model would have 15% accuracy it would be already 30% better thank benchmark but still bad. The problem is, that it is very hard to say where bad accuracy ends and good accuracy starts.

Additionally, I will create confusion matrix in order to see how many false/true positives and negatives are predicted.

II. Analysis

This project will use Bank Marketing Dataset from Machine Learning Repository (link). This dataset represent marketing activities from a bank.

The dataset contains of 20 input variables and out output variable. Variables' descriptions were taken from Machine Learning Repository Website, linked above.

Input variables

Bank client data:
1 - age (numeric)
2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4 - education (categorical:
'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5 - default: has credit in default? (categorical: 'no','yes','unknown')
6 - housing: has housing loan? (categorical: 'no','yes','unknown')
7 - loan: has personal loan? (categorical: 'no','yes','unknown')

Related with the last contact of the current campaign:
8 - contact: contact communication type (categorical: 'cellular','telephone')

9 - month: last contact month of year (categorical: ‘jan’, ‘feb’, ‘mar’, …, ‘nov’, ‘dec’)
10 - day_of_week: last contact day of the week (categorical: ‘mon’,’tue’,’wed’,’thu’,’fri’)
11 - duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y=’no’). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

Other attributes:
12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14 - previous: number of contacts performed before this campaign and for this client (numeric)
15 - poutcome: outcome of the previous marketing campaign (categorical: ‘failure’,’nonexistent’,’success’)

Social and economic context attributes
16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)
17 - cons.price.idx: consumer price index - monthly indicator (numeric)
18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)
19 - euribor3m: euribor 3 month rate - daily indicator (numeric)
20 - nr.employed: number of employees - quarterly indicator (numeric)

Output variable

Output variable:
21 - y - has the client subscribed a term deposit? (binary: ‘yes’,’no’).

Output variable gives an information if the person bought a product (subscribed a term deposit in this dataset). Input variables are actually a description of a person in data, description of campaign, contact time and additional attributes. These attributes are most likely the data which most businesses own and use when advertising their products.

Data Exploration

Dataset contains 21 variables, of which 1 is output variable and 41188 observations.

```
In [28]: dataset_df.shape
Out[28]: (41188, 21)
```

Fig. 1: Dataset shape

Out of 20 input variables in the dataset half is numerical and another one is categorical, stored as objects (strings).

```
In [26]: dataset_df.dtypes

Out[26]: age               int64
         job              object
         marital          object
         education        object
         default          object
         housing          object
         loan             object
         contact          object
         month            object
         day_of_week      object
         duration          int64
         campaign          int64
         pdays             int64
         previous          int64
         poutcome         object
         emp.var.rate    float64
         cons.price.idx  float64
         cons.conf.idx   float64
         euribor3m       float64
         nr.employed     float64
         y                object
         dtype: object
```

Fig. 2: Data types in dataset

```
dataset_df.head()
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	261	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	149	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
2	37	services	married	high.school	no	yes	no	telephone	may	mon	226	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	151	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
4	56	services	married	high.school	no	no	yes	telephone	may	mon	307	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no

Fig. 3: Datset sample

Out of numerical variables, only cons.conf.idx and cons.price.idx look like they would not be skewed. There are also big scale differences between them. Eg. ‘emp.var.rate’ varies from -3.4 to 1.4 and duration from 0 to 4918.

Most of the categorical variables habe 2 to 4 options, apart of job and education as well as date related variables. Some of categorical variables seem to have not categorized data, eg. ‘poutcome’ features has ‘nonexistent’ as the most often input.

```
dataset_df.describe(include='all')
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
count	41188.00000	41188	41188	41188	41188	41188	41188	41188	41188	41188	41188.000000	41188.000000	41188.000000	41188.000000	41188	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188
unique	NaN	12	4	8	3	3	3	2	10	5	NaN	NaN	NaN	NaN	3	NaN	NaN	NaN	NaN	NaN	2
top	NaN	admin.	married	university.degree	no	yes	no	cellular	may	thu	NaN	NaN	NaN	NaN	nonexistent	NaN	NaN	NaN	NaN	NaN	no
freq	NaN	10422	24928	12168	32588	21576	33950	26144	13769	8623	NaN	NaN	NaN	NaN	35563	NaN	NaN	NaN	NaN	NaN	36548
mean	40.02406	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	258.285010	2.567593	962.475454	0.172963	NaN	0.081886	93.575664	-40.502600	3.621291	5167.035911	NaN
std	10.42125	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	259.279249	2.770014	186.910907	0.494901	NaN	1.570960	0.578840	4.628198	1.734447	72.251528	NaN
min	17.00000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.000000	1.000000	0.000000	0.000000	NaN	-3.400000	92.201000	-50.800000	0.634000	4963.600000	NaN
25%	32.00000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	102.000000	1.000000	999.000000	0.000000	NaN	-1.800000	93.075000	-42.700000	1.344000	5099.100000	NaN
50%	38.00000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	180.000000	2.000000	999.000000	0.000000	NaN	1.100000	93.749000	-41.800000	4.857000	5191.000000	NaN
75%	47.00000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	319.000000	3.000000	999.000000	0.000000	NaN	1.400000	93.994000	-36.400000	4.961000	5228.100000	NaN
max	98.00000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4918.000000	56.000000	999.000000	7.000000	NaN	1.400000	94.767000	-26.900000	5.045000	5228.100000	NaN

Fig. 4: Dataset variables description

Exploratory Visualization

Dataset contains numerical and categorical variables. Out of numerical variables, none has normal distributions. Most distributions are positively skewed (age, duration, nr.employed), but some of theme have discrete-like distribution (cons.price.idx, cons.conf.idx)

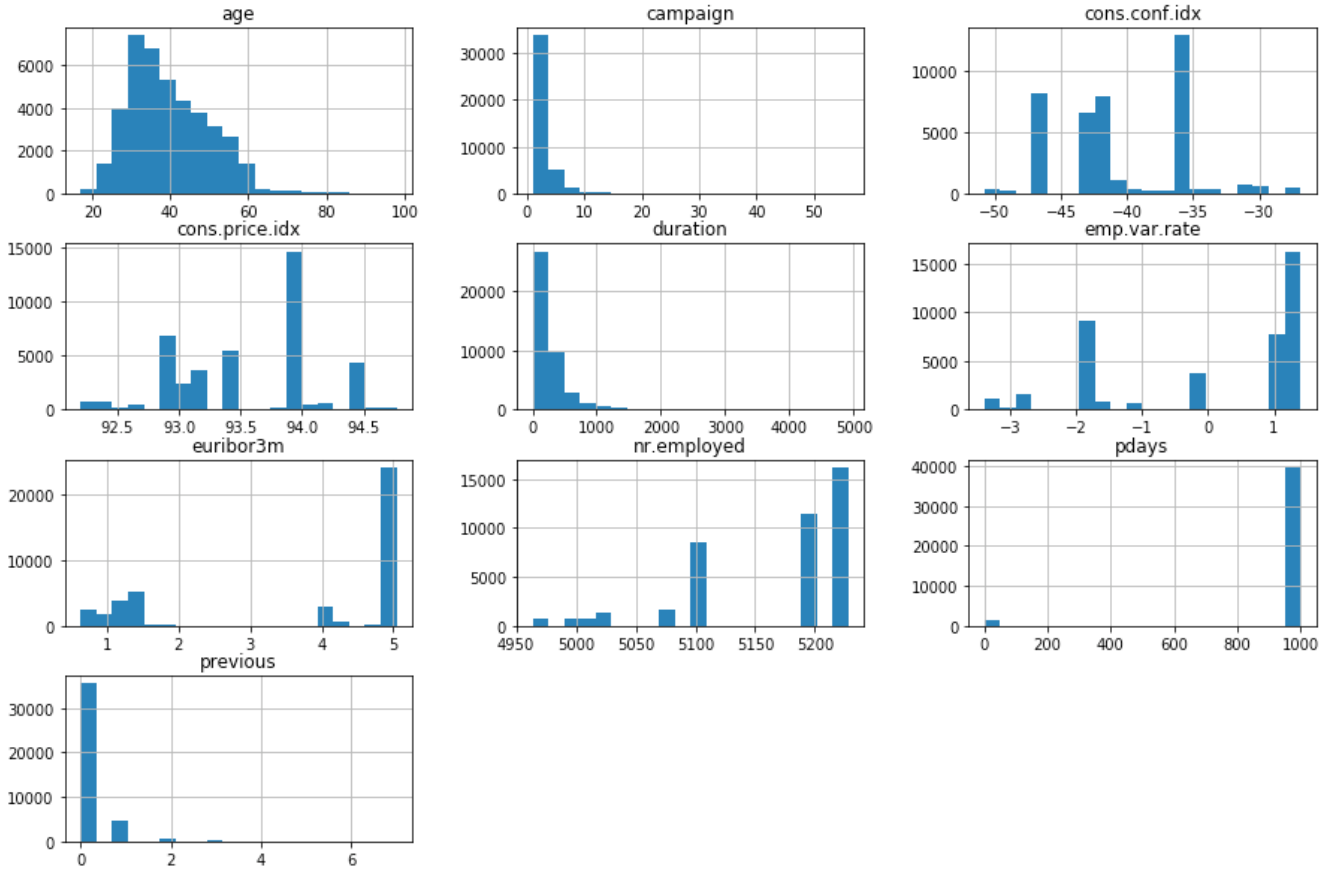

Fig. 5: Distributions of numerical variables

In categorical variables, there are some which are balanced as contact or housing, with housing having very little “unknown” share. There are a few impalanced classes (loan, default) also with very small “unknown” share. Poutcome is imbalanced having “unknown” as the most frequent feature. Marital, Job and Education are only classes with more than 2 labels (when not counting “unknown” as label).

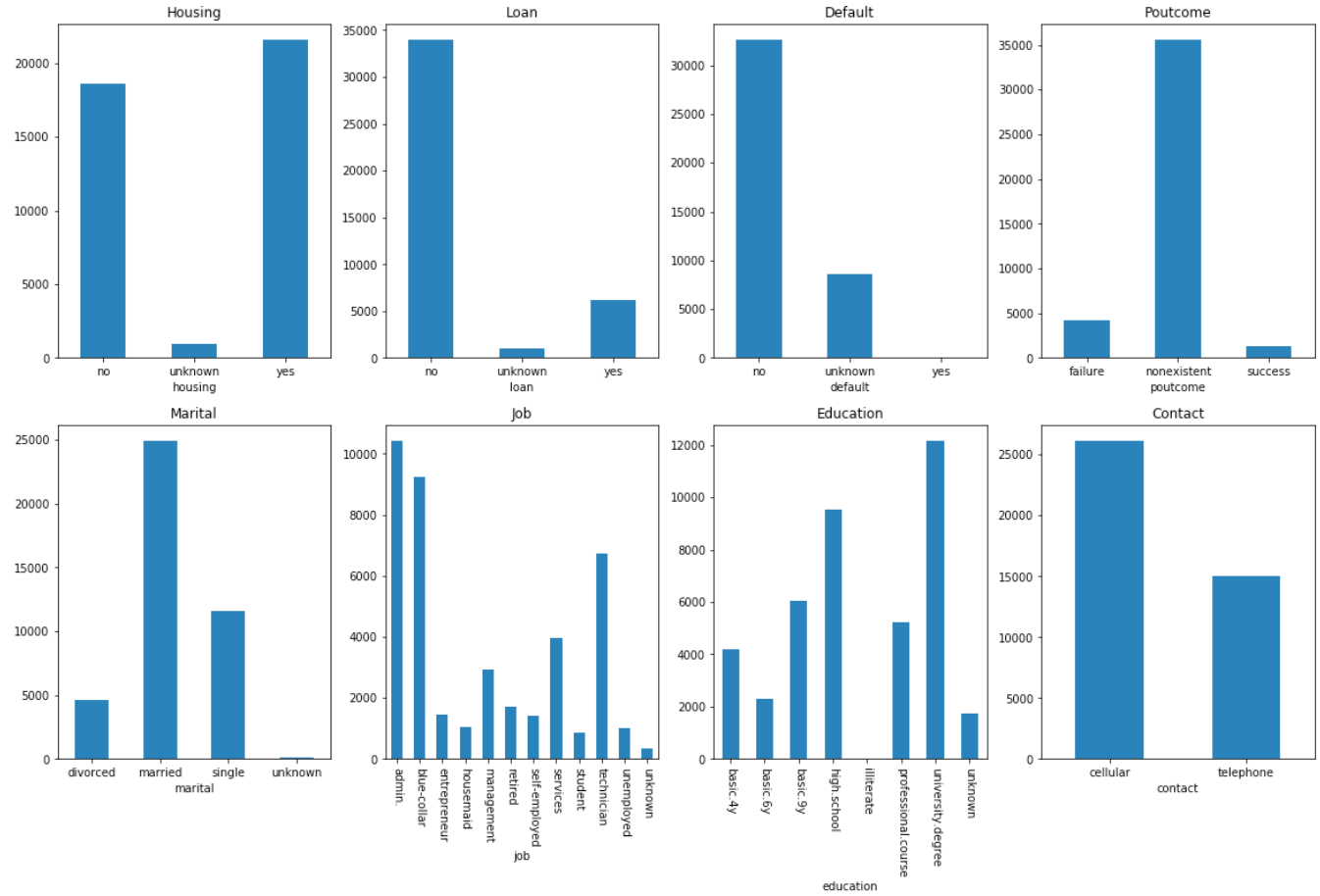

Fig. 6: Distributions of categorical variables

Output variable is an imbalanced two label class with only 11% share of “yes” and 89% “no”.

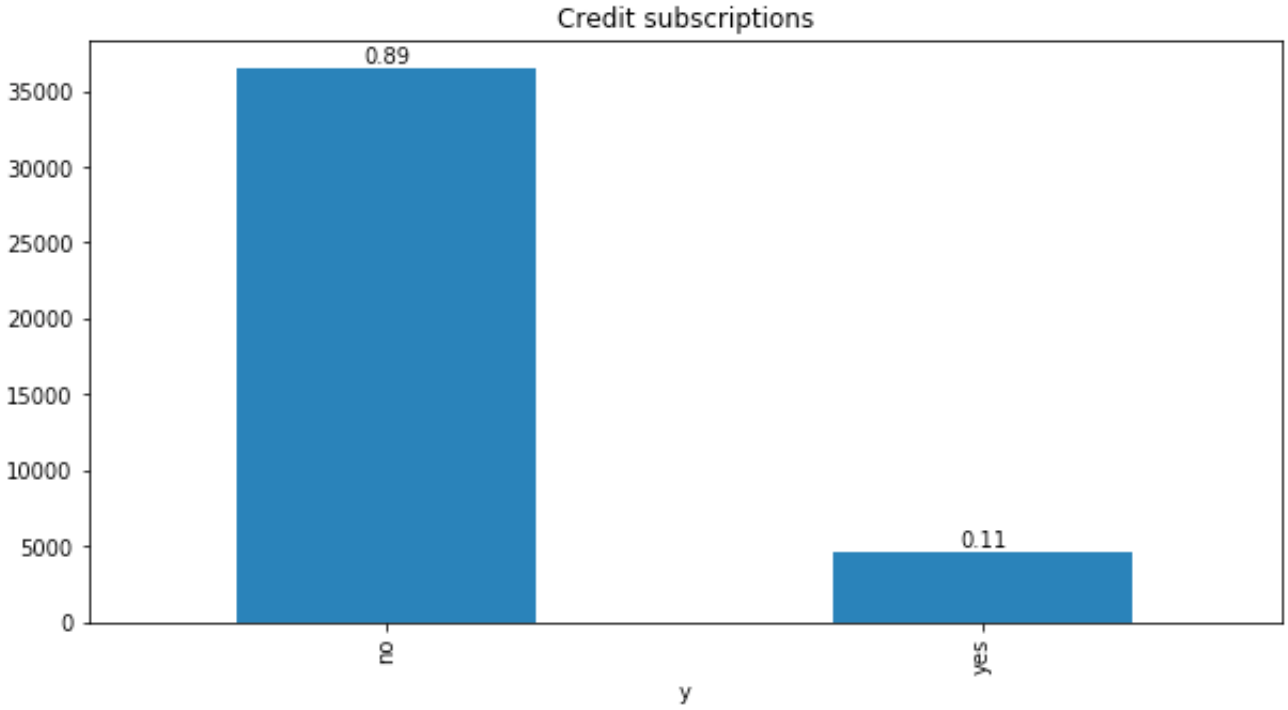

Fig. 7: Distribution of output variable

Algorithms and Techniques

I will choose the best performing classifier out of following:

1. Logistic Regression

Logistic regression is based on a probability of belonging to one or other class. Probability is counted on basis of fit to logistic regression curve (sigmoid function).

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

Fig. 8[4]: Sigmoid function formula

Once data is fitted to the formula, having features coefficients and intercept, every data point can be fitted to the probability function.

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

Fig. 9[4]: Probability function (logistic function) with one dependent variable x

2. Support Vector Machines

Support vector machines construct a hyperplane or set of hyperplanes which divides points in space into two classes. If data is linearly separable, superplane is chosen as line between two closest points from different classes. When data is not linearly separable, so called “kernel trick” can be used in order to make data linearly separable. That means mapping to more-dimensional space.

3. Decision Tree

Decision trees choose sort dependent features based on their importance (having most information about the data) and build from the a tree according to sorting order. It means, that they while working top-down every time choose variable which gives the best split of output variable.

4. Random Forest

Random forest uses decision trees as preliminary for choosing optimal solution. It divides dataset into samples and for every sample builds a decision tree. Additionally, a random subset of features for every sample will be selected. The chosen tree is this, which is built most often.

5. Gaussian NB

Naive bayes apply on Bayes’ theorem with assumption of independence of features. It calculates probability of hypothesis h givenn data d. The model calculates class probabilities (how many % of data belongs to which class) and conditional probabilities, which are probabilities of every feature value for every class value. So in marketing dataset, probability of all 20 input features when given result subscribed or not subscribed.

Benchmark

In order to create a benchmark I assumed, that every contacted client would buy a product. This approach achieved 0.11 accuracy and 0.39 F-2 score.

III. Methodology

Data Preprocessing

Missing data

There is no missing data in this dataset:

```
In [17]: na_df = dataset_transformed_df.isnull().any()

         na_df

Out[17]: age               False
         job               False
         marital           False
         education         False
         default           False
         housing           False
         loan              False
         contact           False
         month             False
         day_of_week       False
         duration          False
         campaign          False
         pdays             False
         previous          False
         poutcome          False
         emp.var.rate      False
         cons.price.idx    False
         cons.conf.idx     False
         euribor3m         False
         nr.employed       False
         y                 False
         dtype: bool
```

Fig. 10: Missing data

But many features have “unknown” which can be considered as missing data. Because I will scale all features I want unknown to be a constant between “yes” and “no”. I will change all “unknown” or “nonexistent” to 0 and accordingly, “yes” or “success” to 1 and “no” or “failure” to -1.

```
In [18]: dataset_transformed_df['housing'] = dataset_transformed_df['housing'].map({'yes': 1, 'no': -1, 'unknown': 0})
         dataset_transformed_df['loan'] = dataset_transformed_df['loan'].map({'yes': 1, 'no': -1, 'unknown': 0})
         dataset_transformed_df['default'] = dataset_transformed_df['default'].map({'yes': 1, 'no': -1, 'unknown': 0})
         dataset_transformed_df['poutcome'] = dataset_transformed_df['poutcome'].map({'success': 1, 'failure': -1, 'nonexistent': 0})

In [19]: dataset_transformed_df.head()
```

Out[19]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	4.043051	housemaid	married	basic.4y	-1	-1	-1	telephone	may	mon	5.568345	0.693147	999	0	0	1.1	93.994	-36.4	4.857	8.554874	0
1	4.060443	services	married	high.school	0	-1	-1	telephone	may	mon	5.010635	0.693147	999	0	0	1.1	93.994	-36.4	4.857	8.554874	0
2	3.637586	services	married	high.school	-1	1	-1	telephone	may	mon	5.424950	0.693147	999	0	0	1.1	93.994	-36.4	4.857	8.554874	0
3	3.713572	admin.	married	basic.6y	-1	-1	-1	telephone	may	mon	5.023881	0.693147	999	0	0	1.1	93.994	-36.4	4.857	8.554874	0
4	4.043051	services	married	high.school	-1	-1	1	telephone	may	mon	5.730100	0.693147	999	0	0	1.1	93.994	-36.4	4.857	8.554874	0

Fig. 11: Replacing “unknonwn” and “nonexistent” with 0

Additionally, I will insert dummies for categorical variables.

```
In [20]: dummies = pd.get_dummies(dataset_transformed_df[['job', 'marital', 'education', 'contact', 'month', 'day_of_week']])

         dataset_transformed_df = dataset_transformed_df.join(dummies)

         dataset_transformed_df = dataset_transformed_df.drop(['job', 'marital', 'education', 'contact', 'month', 'day_of_week'], axis=1)

         dataset_transformed_df.head()
```

Out[20]:

y	job_admin.	job_blue-collar	job_entrepreneur	job_housemaid	job_management	job_retired	job_self-employed	job_services	job_student	job_technician	job_unemployed	job_unknown	marital_div
0	0	0	0	1	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	1	0	0	0	0	
0	0	0	0	0	0	0	0	1	0	0	0	0	
0	1	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	1	0	0	0	0	

Fig. 12: Inserting dummies

Skewed distributions

As many distributions are skewed, I will transform then using log transformation, log(1+x).

```
dataset_transformed_df['age'] = np.log1p(dataset_transformed_df['age'])

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(20,10))

ax1 = dataset_transformed_df['age'].hist(ax=axes[0])
ax1.title.set_text('"age" transformed')

ax2 = dataset_df['age'].hist(ax=axes[1])
ax2.title.set_text('"age" original data')
```

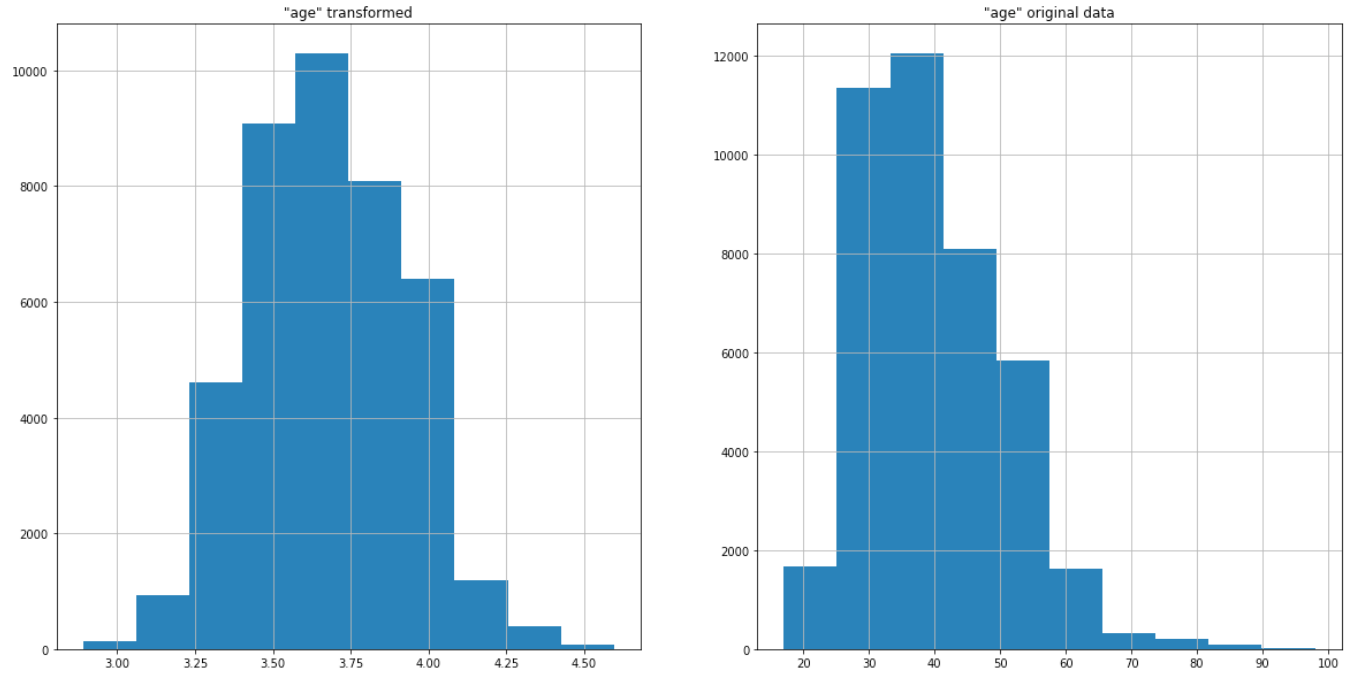

Fig. 13: Transformed age feature

```
dataset_transformed_df['duration'] = np.log1p(dataset_transformed_df['duration'])

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(20,10))

ax1 = dataset_transformed_df['duration'].hist(ax=axes[0])
ax1.title.set_text('"duration" transformed')
ax2 = dataset_df['duration'].hist(ax=axes[1])
ax2.title.set_text('"duration" original data')
```

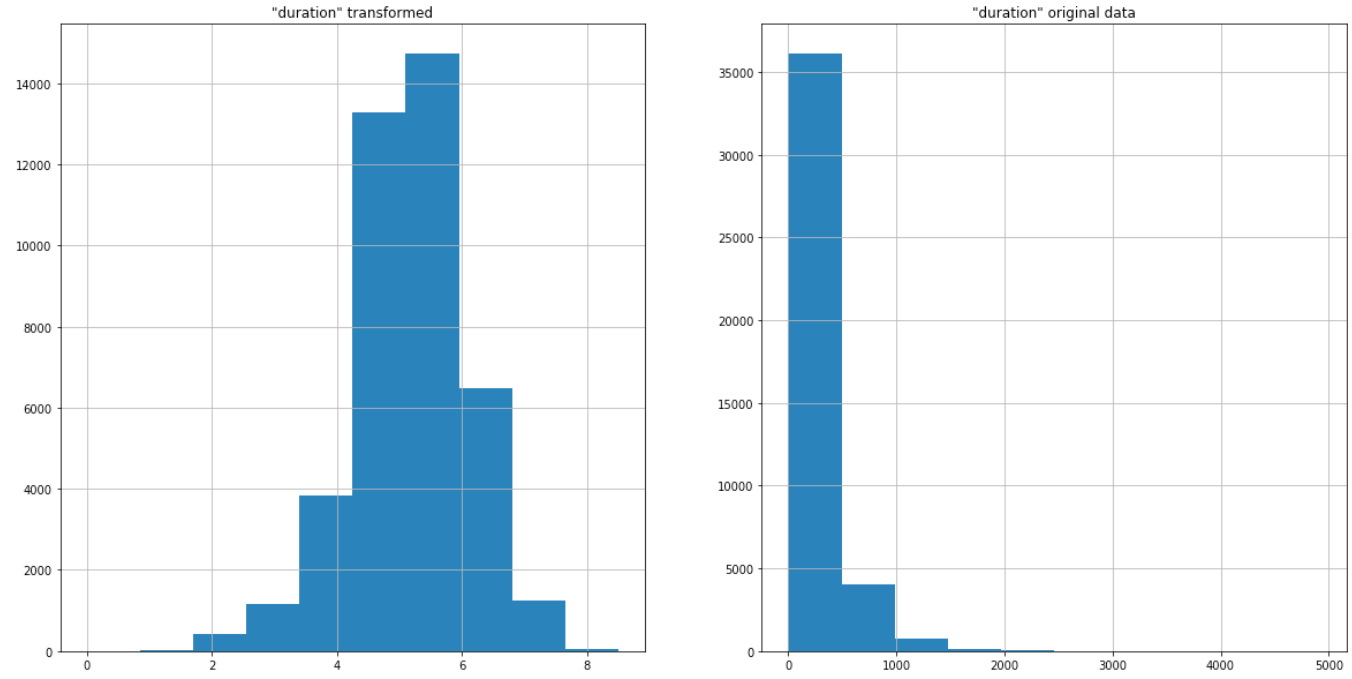

Fig. 14: Transformed duration feature

```
dataset_transformed_df['campaign'] = np.log1p(dataset_transformed_df['campaign'])

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(20,10))

ax1 = dataset_transformed_df['campaign'].hist(ax=axes[0])
ax1.title.set_text('"campaign" transformed')
ax2 = dataset_df['campaign'].hist(ax=axes[1])
ax2.title.set_text('"campaign" original data')
```

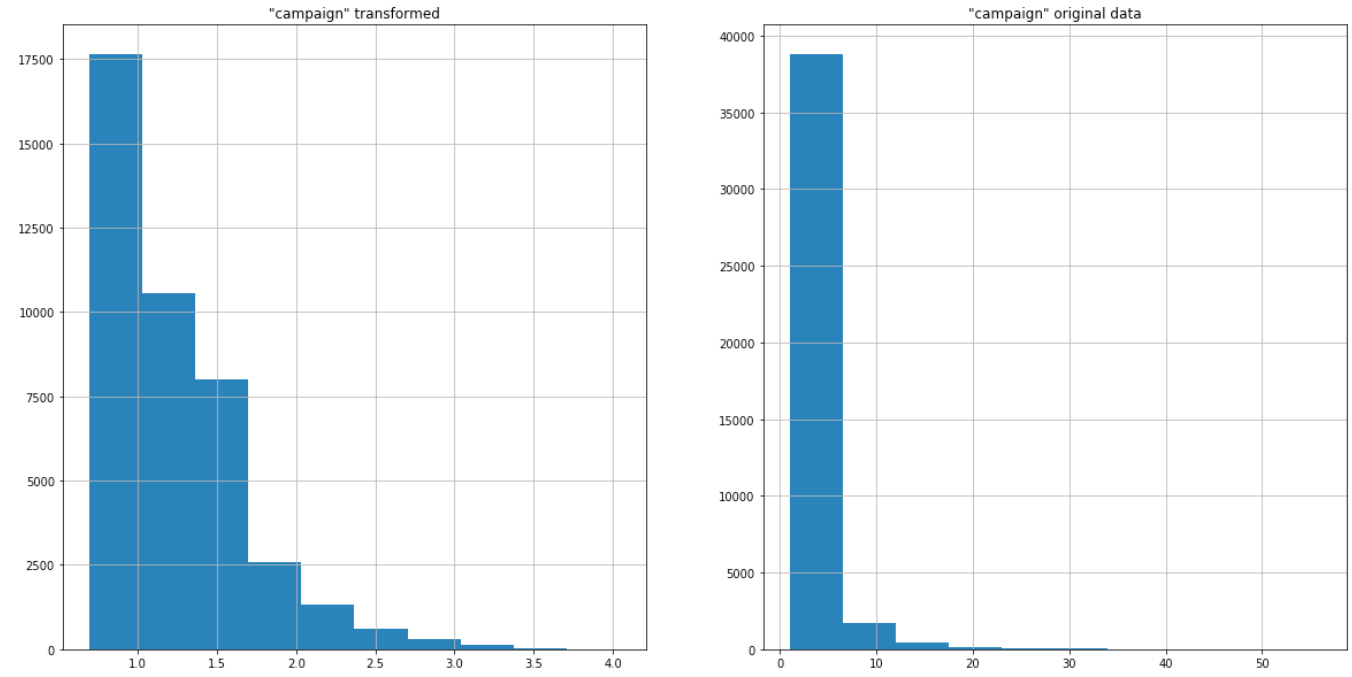

Fig. 15: Transformed campaign feature

Implementation

Data preprocessing

All data was preprocessed, as described earlier before starting any model evaluation or fitting. Then, before splitting data info test and training data, it had to be under-sampled in order to minimize classes imbalance.

Random Under-Sampling

Because this dataset is very imbalanced (“yes” output class is only 11% of whole dataset) and also pretty large (41k observations), I will perform under-sampling in order to train and test on more balanced dataset. That means, that I will randomly choose 30% of data with “no” output and combine in with all data with “yes” output before split dataset into training and testing. It causes dataset to be smaller, only with 15k observations, but it also more balanced, with 30% share of “yes” class.

```
In [4]: dataset_df.shape
Out[4]: (15604, 21)
```

Fig. 16: Dataset shape after under-sampling

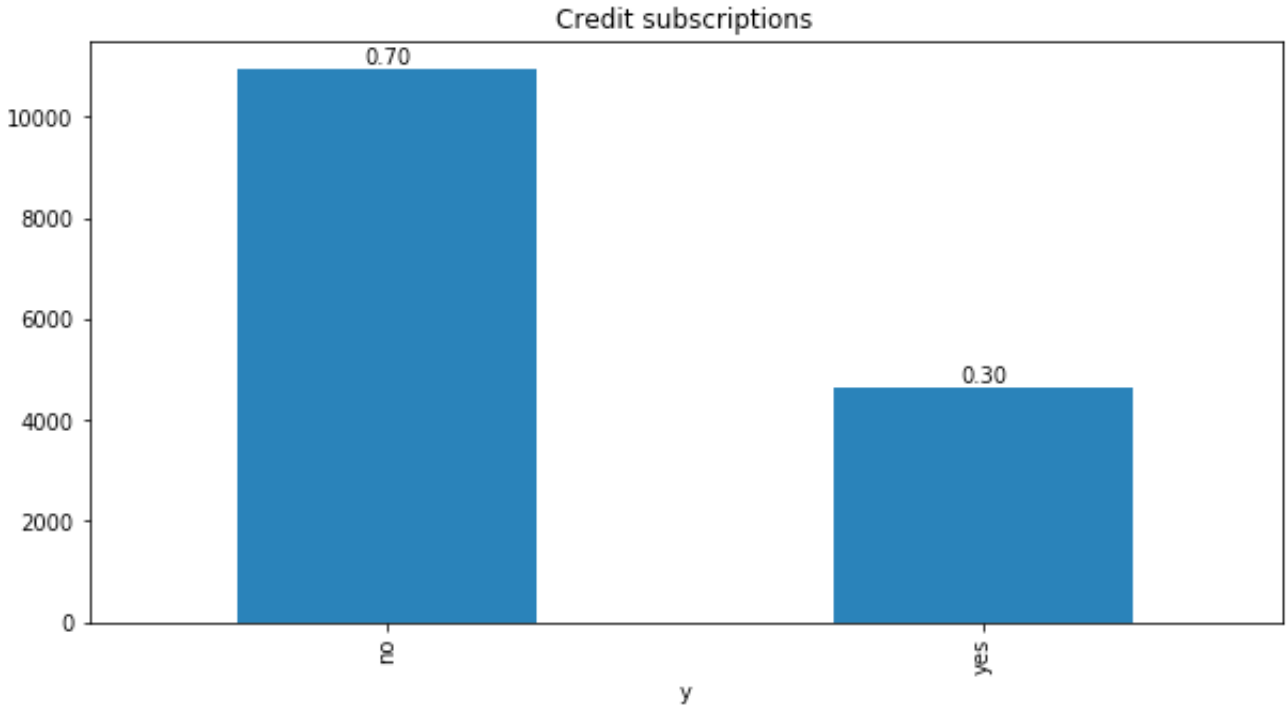

Fig. 17: Classes share after under-sampling

Random under-sampling was done 10 times every time with another split. This gives the possibility to test the model robustness on different data splits and decide which model performs the best.

Model fitting

Model fitting and predicting, but also splitting data into test and training parts was done in sklearn. Sklearn.SchuffleSplit is method, which is randomly splittind data into two datasets - one for training and one for testing. In this project 75% of data was used for training the algorithm and 25% for testing. That means,

that when algorithm is trained, it “sees” only 75% of data (it is the same data for every algorithm). Every model has it’s implementation in sklearn library:

Model	Sklearn implementation
Logistic Regression	LogisticRegression
Gaussian Naive Bayes	GaussianNB
Support Vector Machines	svm
Decision Tree	DecisionTreeClassifier
Random Forest	RandomForestClassifier

Fig. 18: Sklearn classification models implementations

For every model, .fit() method is used to fit the classifier to training data and .predict() method to make prediction on test data after model has been trained. Fitting model means actually to train it. After successful training, the trained classifier is used to make prediction on the test set. These predictions are compared with real testing data and on this basis metrics are calculated. Metrics are also predefined in sklearn, from which F-Beta score, accuracy, precision and recall were used.

Choosing best model

Because running time of testing all possible parameters for every model is very time inefficient and cannot be done for many different splits, best model is chosen without doing any hyperparameter tuning. For every of 10 different under-samplings, there are 10 random training and test splits done. On every of these 100 different splits, each of 5 models is trained and measured. Model evaluation will be done for average and median of all 100 performances. Out of all models, SVMs have performed the best in terms of F2-Score (fbeta) score, accuracy and recall. Precision was not the best of all models, but as written above, precision is not the main measure in this project.

Model	mean F2-Score	mean Accuracy	mean Recall	mean Precision
Logistic Regression	0.756	0.867	0.748	0.786
Gaussian Naive Bayes	0.534	0.787	0.506	0.695
Support Vector Machines	0.806	0.875	0.814	0.775
Decision Tree	0.725	0.839	0.724	0.730
Random Forest	0.745	0.862	0.735	0.786

Fig. 19: Mean metrics of all algorithms

According to student T-Test, SVM F-2 Score (sample 2) was significantly bigger in t-test than second biggest score, which came from logstic regression (sample 1).

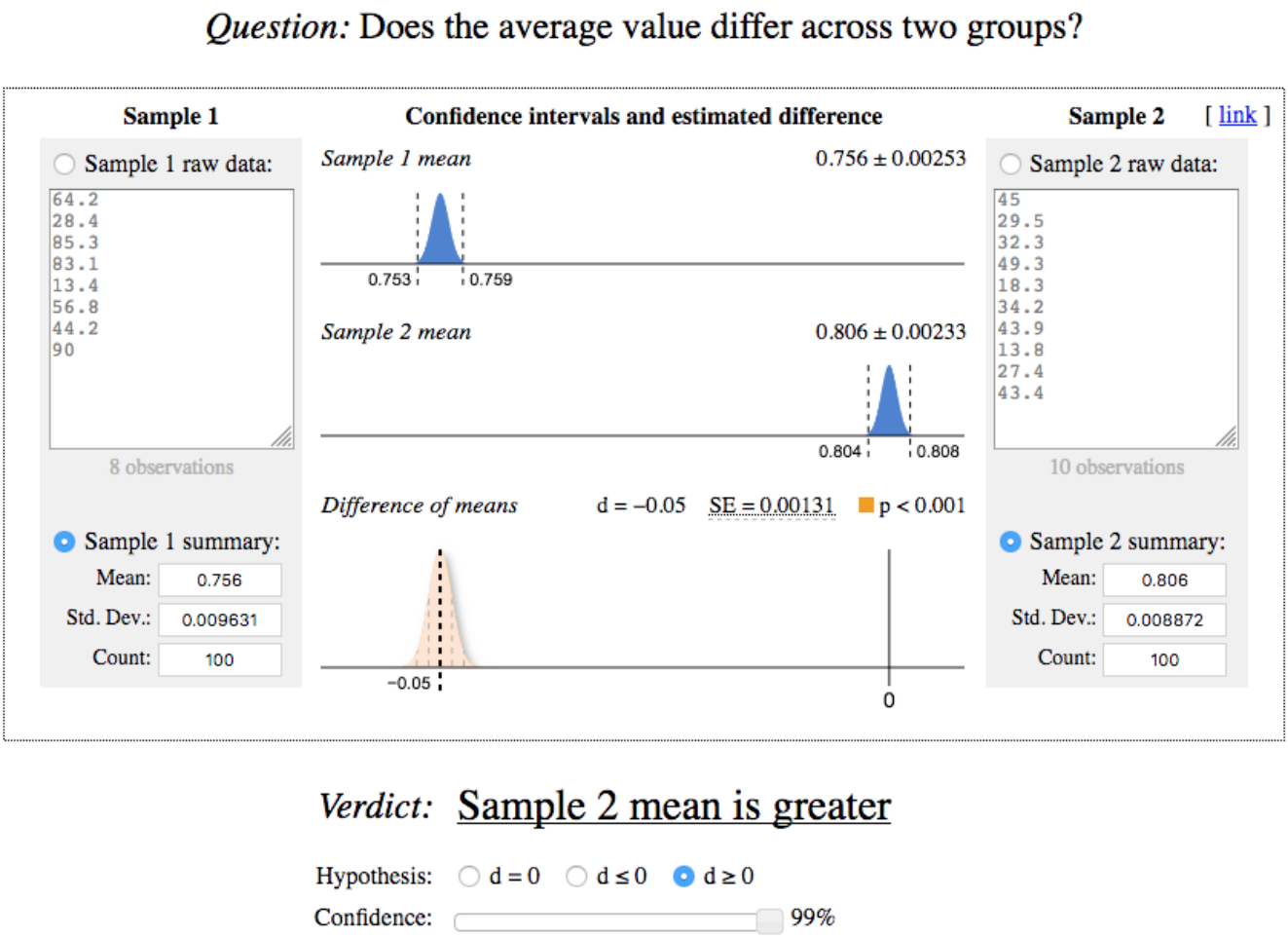

Fig. 20: Student's T-Test for logistic regression and SVMs

On this basis, SVMs is chosen to do further parameter tuning.

Refinement

In support vector machines, two main hyperparameters can be tuned: C and kernel. Kernel means how the data is splitted in the space - if plane should be straight line, polynomial or RBF, which plotted on a 2 dimensional spaces look something like circle.

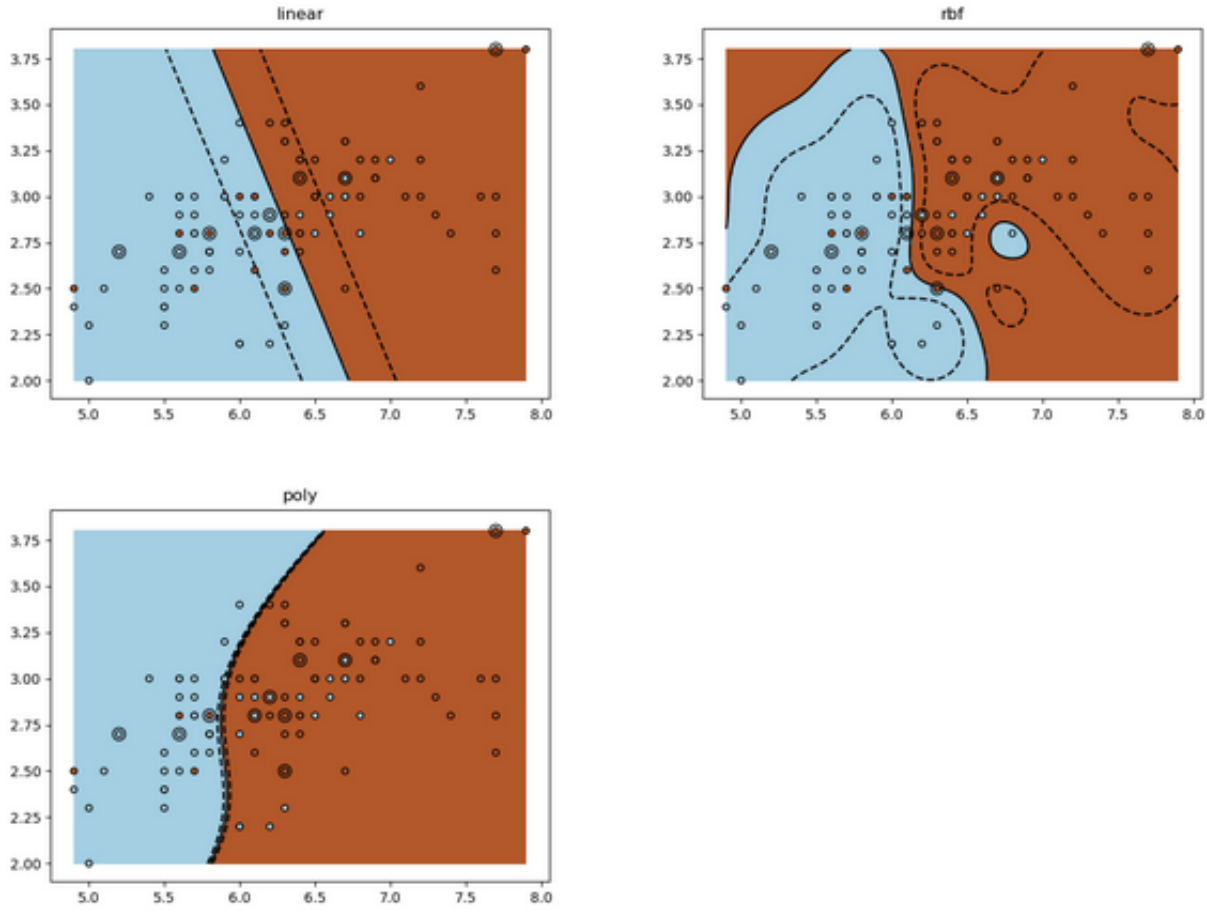

Fig. 21[5]: SVM kernel types

C parameter tells SVM how much sensitive it should be for missclasifying points. The bigger C is, the more points should be classified correctly. Keeping that in mind, bigger C runs into risk of overfitting the data.

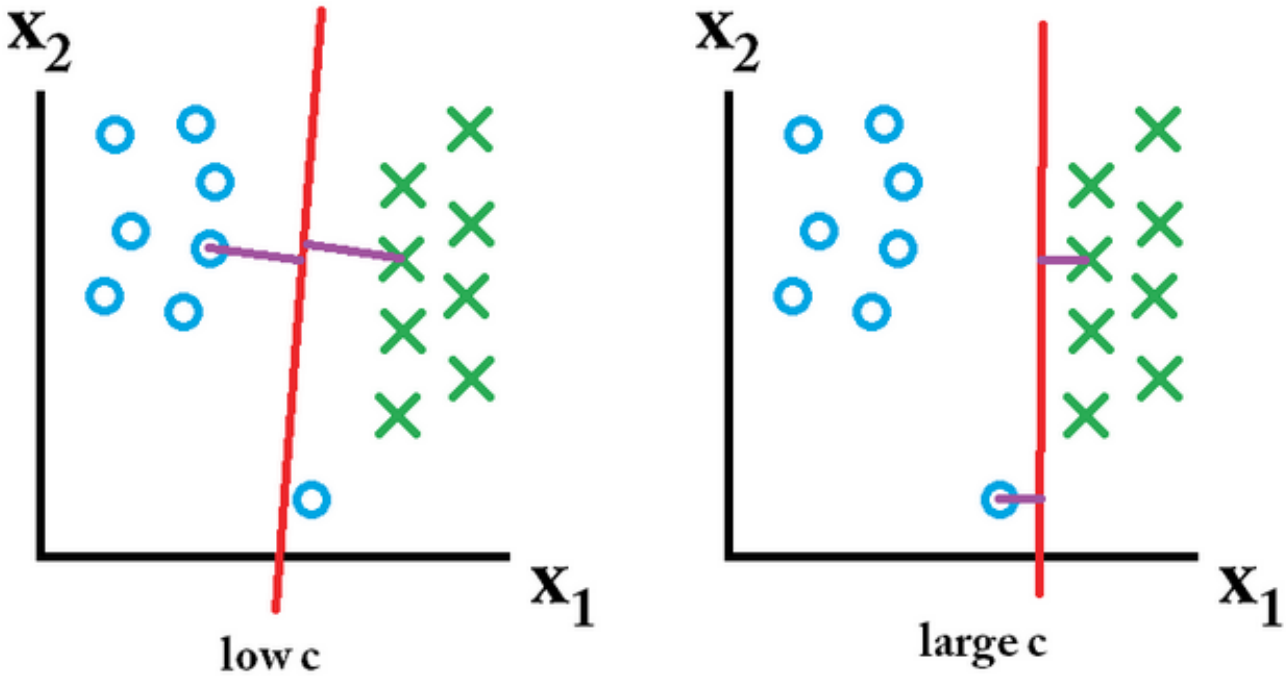

Fig. 22[6]: SVM C parameter

Additionally, for kernel function, there is a gamma parameter, which indicates how much important the variance is. That means, for small gamma, two points can be further away to be considered as similar, but for

large gamma, points have to be close to each other to be considered similar.

I did hyperparameter tuning using GridSearchCV with 2 shuffle splits and 3 folds for each split. I used this example for my implementation (link). Because recall is most important factor for me, which is also used for counting f2-score, I optimized GridSearchCV on recall.

Following hyperparameters and their combinations were tested (values in columns are combined with values in another columns, but values in rows not):

Kernel	C	gamma
RBF	1, 10, 100, 1000	1, 0.1, 0.01, 0.001
linear	1, 10, 100	-

Fig. 23: Tested SVM hyperparameters

Best parameters are listed above and under “Grid scores on development set” are listed all parameters:

```
Best parameters set found on development set:

{'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}

Grid scores on development set:

0.081 (+/-0.009) for {'C': 1, 'gamma': 1, 'kernel': 'rbf'}
0.815 (+/-0.011) for {'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
0.794 (+/-0.014) for {'C': 1, 'gamma': 0.01, 'kernel': 'rbf'}
0.656 (+/-0.020) for {'C': 1, 'gamma': 0.001, 'kernel': 'rbf'}
0.131 (+/-0.016) for {'C': 10, 'gamma': 1, 'kernel': 'rbf'}
0.770 (+/-0.026) for {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
0.803 (+/-0.015) for {'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}
0.782 (+/-0.007) for {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
0.131 (+/-0.019) for {'C': 100, 'gamma': 1, 'kernel': 'rbf'}
0.729 (+/-0.022) for {'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}
0.784 (+/-0.006) for {'C': 100, 'gamma': 0.01, 'kernel': 'rbf'}
0.802 (+/-0.012) for {'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
0.131 (+/-0.020) for {'C': 1000, 'gamma': 1, 'kernel': 'rbf'}
0.729 (+/-0.023) for {'C': 1000, 'gamma': 0.1, 'kernel': 'rbf'}
0.743 (+/-0.024) for {'C': 1000, 'gamma': 0.01, 'kernel': 'rbf'}
0.797 (+/-0.019) for {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
0.768 (+/-0.019) for {'C': 1, 'kernel': 'linear'}
0.746 (+/-0.022) for {'C': 10, 'kernel': 'linear'}
0.737 (+/-0.007) for {'C': 100, 'kernel': 'linear'}
```

Fig. 24: Optimized SVM parameters

When using best parameter configuration, following scores were achieved (for 100 training / testing runs):

	mean F2-Score	mean Accuracy	mean Recall	mean Precision

Before hyperparameter tuning	0.806	0.875	0.814	0.775
After hyperparameter tuning	0.808	0.874	0.818	0.769

Fig. 25: Optimized SVM parameters

Optimized model has 0.002 better F2-Score than not optimized - from 0.806 to 0.808, so the optimizations will be used for further model evaluation.

IV. Results

Model Evaluation and Validation

Chosen model was support vector machines with hyperparameters described in refinement section. The goal of the model was to achieve higher F2-score than benchmark. SMVs performed best out of 5 tested models and additionaly their performance could be slightly improved through hyperparameter tuning. Because model training was run 100 times, final results will be evaluated based on mean and average performances.

In order to achieve robustness, model was run 100 times with different splits and different under-samplings. This caused small fluctuation, with minimum F2-score of 0.79 and maximum 0.83. However, it was still the best out of all tested models.

Justification

SVM achieved 0.807 F2-score which is 208% more than benchmark. It also achieved higher accuracy and precision, but smaller recall. Reason for achieving smaller recall is that in the benchmark we assumed that all people which are contacted will subscribe to a product, which causes that all positives are predicted correctly. In layman terms, it is more the reason that benchmark had very high recall than the SVMs are performing badly. Minimal F2-Score of SVMs before tuning was on the same level as best score from second classifier - logistic regression.

Fig. 26: Final results model vs. benchmark

Fig. 27: Final results model (means) vs. benchmark

SVM were significantly better than benchmark according also to two sample mean t-test. In figure below, sample one is F2-Score of SVMs and sample 2 is benchmark F2-Score.

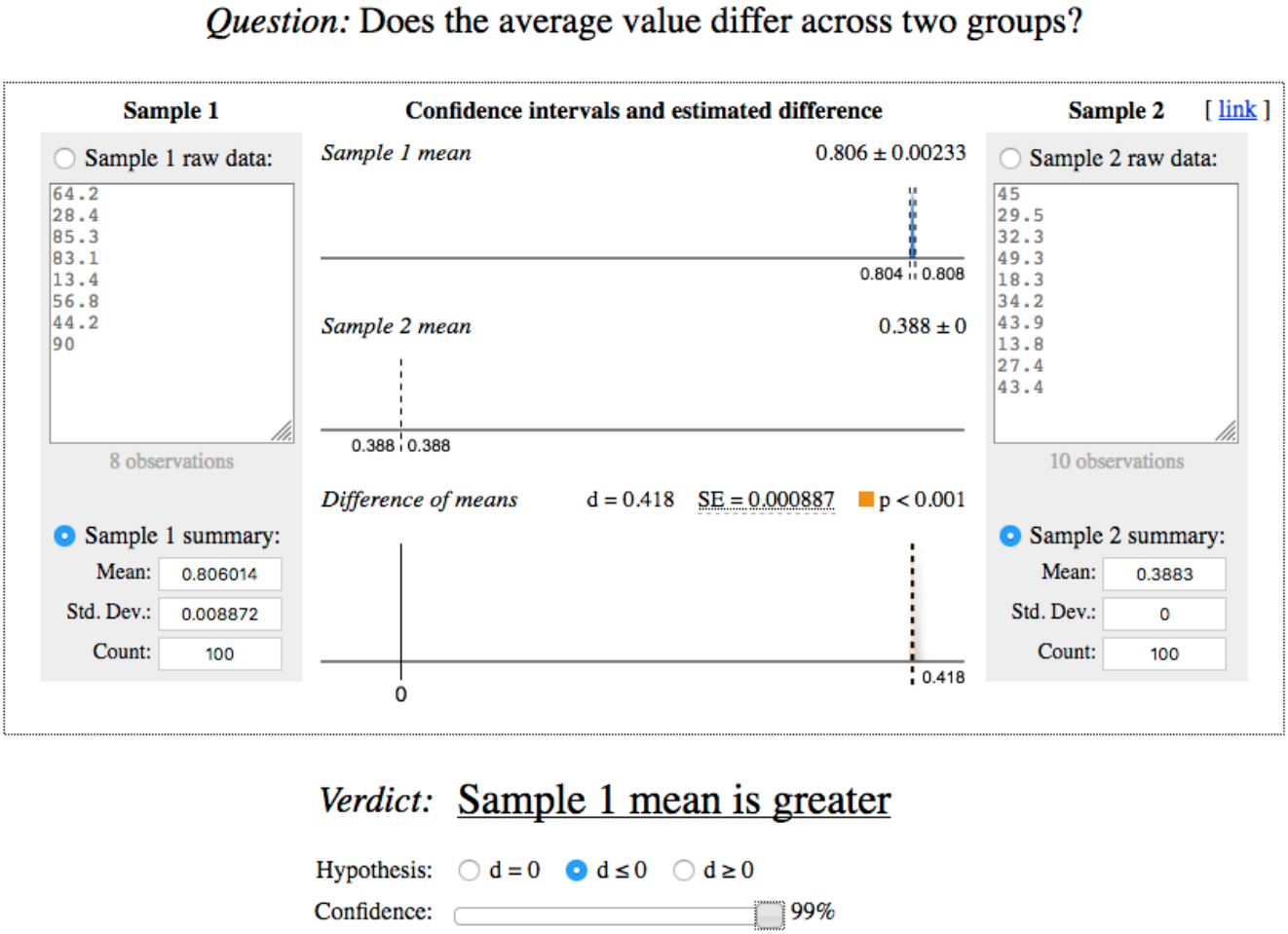

Fig. 28[7]: T-Test result for SVMs score vs. benchmark

V. Conclusion

Free-Form Visualization

This project would be used in marketing department in order to decide which potential customers to contact. Because sales people are contacting customers through telephone, it takes time and money to contact every customer which will not subscribe to a deposit. Proposed algorithm has 0.82 recall, which means that if this algorithm would be used to choose which potential customers should be contacted, only 82% of customers who bought the product would buy it, because other 18% of them would not be contacted at all. It can mean, that it would decrease sales, but there is another side of it. Algorithms' recall on potential client, who didn't buy the deposite is 0.88. That means that usage algorithm would reduce unsuccessfull contacts to 12%. Keeping in mind, that 89% of people in this dataset didn't subscribe to a deposit, 80% less people would have to be contacted, what would cause only 12% smaller order intake.

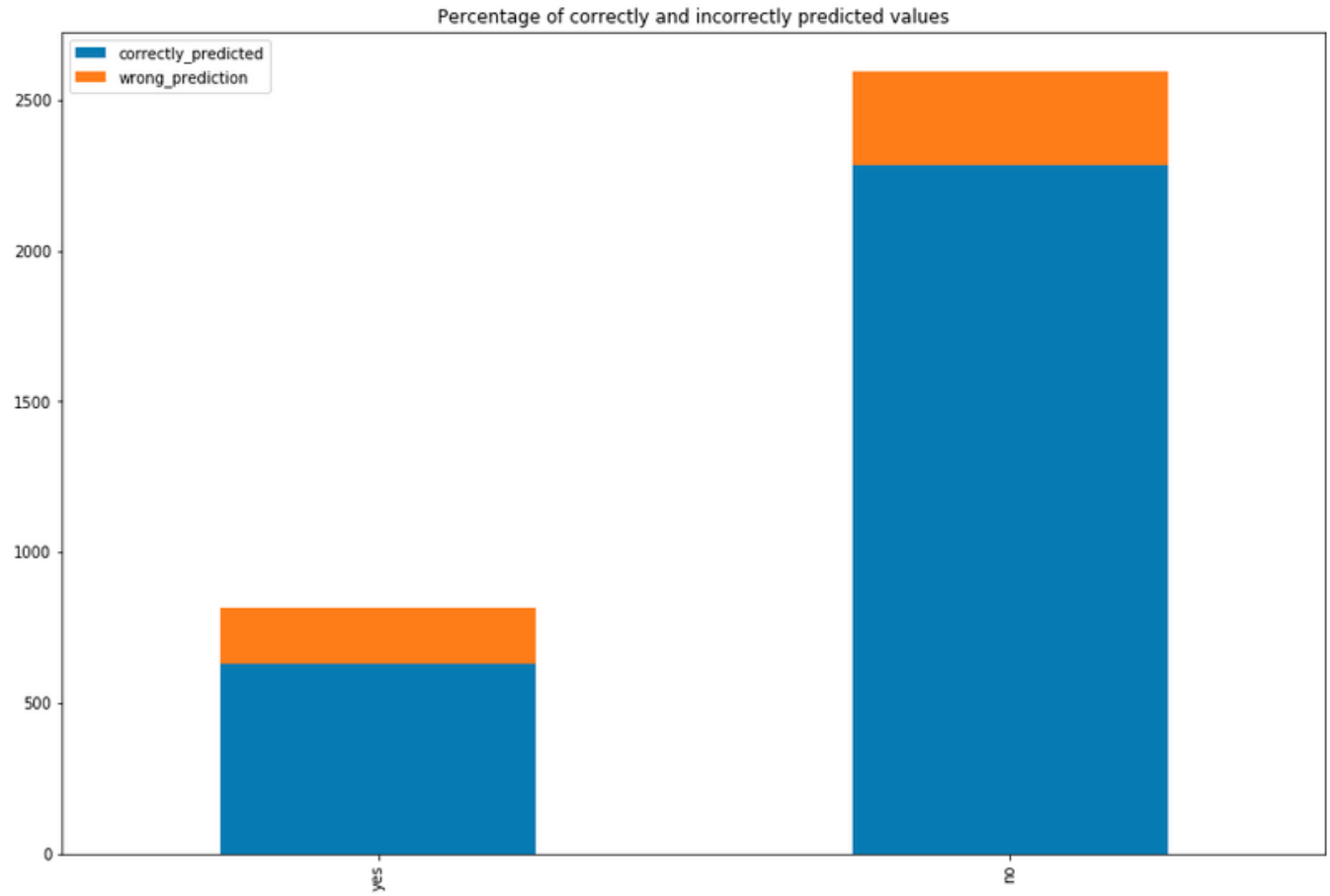

Fig. 29: T-Test result for SVMs score vs. benchmark

Reflection

The process of creating this solution was following marketing actions, when there is a need to optimize marketing performance.

1. There was a dataset of marketing activities and their results (real data obtained from repository).
2. Goal was defined to gel as high F2-Score as possible, but at least higher than benchmark.
3. Data was investigated and described.
4. Benchmark metrics were set.
5. Data was preprocessed and accordingly manipulated.
6. Optimal algorithm for this task was chosen.
7. Classifier was tuned and trained on the data.
8. Classifier was tested for robustness.

Because data was imbalanced, the most difficult part was to preprocess the data so that result is highest achievable, but not biased and robust. It was achieved with under-sampling the dataset, which was possible only because dataset was large enough.

Improvement

As this is general implementation, it could be improved for any specific purpose. Some example

improvements are:

1. Using training and prediction time as one of choosing criteria for final algorithm, as it might be used for api where marketeers can enter data of new potential client and get real time answer if it is worth to contact him/her.
2. Under-sampling could be suplemented with generating synthetic positive data, which would lead to usage of whole dataset and probably better predictions.
3. Project could serve generally as advisory about which clients should be contacted, therefore PCA (principal component analysis) or an algorithm with feature importances could be used (as decision trees).
4. Finally, supplementary to first point, it can be used in some kind of application wrapped around it so that it is real-life ready to use solution.

---

[1] Pål Sundsøy, Johannes Bjelland, Asif M Iqbal, Alex Sandy Pentland, Yves-Alexandre de Montjoye. *Big Data-Driven Marketing: How machine learning outperforms marketers' gut-feeling.* In *Social Computing, Behavioral-Cultural Modeling & Prediction Lecture Notes* in *Computer Science* Volume 8393. 2014, pp. 367-374.

[2] Dirk Thorleuchter, Dirk Van den Poel, Anita Prinzie. *Analyzing existing customers' websites to improve the customer acquisition process as well as the profitability prediction in B-to-B marketing.* In *Expert Systems with Applications* Volume 39(3). February 2012, pp. 2597-2605.

[3] Bo Pang, Lillian Lee, Shivakumar Vaithyanathan. *Thumbs up? Sentiment Classification using Machine Learning Techniques* In *Proceedings of EMNLP*. 2002, pp. 79–86.

[4] https://en.wikipedia.org/wiki/Logistic_regression

[5] http://scikit-learn.org/stable/auto_examples/exercises/plot_iris_exercise.html#sphx-glr-auto-examples-exercises-plot-iris-exercise-py

[6] https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel

[7] http://www.evanmiller.org/ab-testing/t-test.html