ADACTA

# Workshop

## Client behavior prediction: A Machine Learning Challenge

# Goal of the workshop

- To introduce the problem
- Get a quick overview of the ML pipeline
- Explain the requirements

# Outline of the workshop

- Introduction to client behaviour prediction
  - Problem, solution and evaluation from the domain experts‘ point of view
- Machine learning
  - Business process & AI
  - ML overview
  - ML Pipeline
- ML for client behaviour prediction
  - Requirements for the competition
- Conclusion
- Q&A

# Client behaviour prediction

# Client behaviour

Raiffeisen
BANK

# Machine learning

# Business process & AI

- Companies collect and store data for ages
  - Keep track of the balance (client balance, storage balance, …)
  - Regulations (financial, …)
  - …
- The value of data was redefined by the advancements in AI and computational power

- Empower management to make better decisions
- Detect anomalies
  Predict anomalies
- Reorganise logistics in real time
- Optimise quality of the products
- Identify opportunities
- Predict client behaviour
- Forecast future outcomes

# Machine-learning overview

- Learning techniques (according to the amount of labelled data)
  - Unsupervised learning (no labelled data)
  - Supervised learning (labelled data)
  - Semi-supervised learning (labelled & unlabelled data)

The data is labelled with
- Y – premature repayment
- N – regular repayment

- Two types of problems
  - Classification (categorical target)
  - Regression (continuous target)

The target is categorical
- Y – premature repayment
- N – regular repayment

- Tasks
  - Prediction → Historic data is used to predict future (predict future from current example)
  - Recognition/Detection → Historic data is used to learn patterns of interest (recognise current example)
  - Estimation → Estimate a value either in a future or of the current example

# The ML pipeline

### Data source

Labelled/Unlabelled dataset

Additional dataset

Expert knowledge

### Data fusion

Structured or unstructured data

Difference in raw data
- Frequency
- Sampling rate
- Modalities

Visualisation of data

**Result:**
**Synchronised raw data**

### Data pre-processing

Raw data segmentation

Data cleaning

Window size selection
- Time related?
- Sliding or fixed?
- Dynamic or static segmentation

**Result:**
**Raw data in windows**

### Data processing

Feature extraction

- Semantic features
- Statistical features
- Historic features
- Demographic features
- Antrophomorphic features
- …

Visualisation of extracted features

**Result:**
**Feature vectors**

### Training & Evaluation

Feature selection

ML algorithm selection
- Single ML algorithm
- Ensembles
- Meta ML
- Tune ML parameters

Evaluation:
- x-fold-cross
- leave-one-entitiy-out
- Test set

**Result:**
**Evaluated ML model or method**

### Validation

ML model or method is validated on unseen dataset
- Overfitting
- Cheating

**Result:**
**Realistic performance score**

[1] Fayyad et al. (1996). From Data Mining to Knowledge Discovery in Databases. AI Magazine, 17 (3), 37. doi:10.1609/aimag.v17i3.1230

# Machine learning pipeline Use Case

# Workshop use case

- Labelled datasets:
  - Client infromation
  - Loan information
  - *Client transaction history*



- Goal: **Recognise Bad Loans**
  - Use historic data to learn patterns of good and bad loan
  - Train a model
  - Use model to classify loan data

# Data fusion

- Low-level data fusion combines several sources of raw data to produce new raw data
  - We do not want to lose any data!

## Use case:

- Fuse the data of the two databases
  - Client information – demographic data
  - Loan information today – single example

| date | loanID | clientID | demographic | $\text{Loan\_information}_{\text{date}}$ | $\text{Loan\_quality}_{\text{date}}$ |
|---|---|---|---|---|---|

# Data fusion

- Low-level data fusion combines several sources of raw data to produce new raw data
  - We do not want to lose any data!

## Use case:

- Fuse the data of the two databases
  - Client information – demographic data
  - Loan information today – single example
  - *Client transaction history*

If we had a timeseries of client transaction histrory

Missing values

| | | | | | | |
|---|---|---|---|---|---|---|
| $date_1$ | loanID | clientID | demographic | $Transactions_{date1}$ | $Loan\_information_{date1}$ | $Loan\_quality_{date1}$ |
| $date_2$ | loanID | clientID | demographic | $Transactions_{date2}$ | $Loan\_information_{date2}$ | $Loan\_quality_{date2}$ |
| $date_3$ | loanID | clientID | demographic | $Transactions_{date3}$ | $Loan\_information_{date3}$ | $Loan\_quality_{date3}$ |

…

| | | | | | | |
|---|---|---|---|---|---|---|
| $date_n$ | loanID | clientID | demographic | $Transactions_{daten}$ | $Loan\_information_{daten}$ | $Loan\_quality_{daten}$ |

# Data pre-processing

- One example per loan (limited by the loan information) ← window size

| date | loanID | clientID | demographic | $\text{Loan\_information}_{date}$ | $\text{Loan\_quality}_{date}$ |
|---|---|---|---|---|---|

- Clean data
  - Missing values – Imputation vs. removing
  - Anomalies / Outliers

- Encoding categorical data

[2] Saar-Tsechansky M. et al. (2007) Handling Missing Values when Applying Classification Models. J. Mach. Learn. Res.

[3] How to Handle Missing Data, https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4

# Data pre-processing

## 2 - Data preprocessing

- Rename/Drop columns
- Do something with the missing values
- Check if any outliers exist - remove those

[3]:
```
df.drop(['Unnamed: 0'], axis=1, inplace=True)
df.head()
```

[3]:

| | loan_amount | funded_amount | investor_funds | term | interest_rate | installment | grade | sub_grade | emp_length | home_ownership | ... | application_type | acc_now_delinq |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 5000 | 5000 | 4975.0 | 36 months | 10.65 | 162.87 | B | B2 | 10+ years | RENT | ... | INDIVIDUAL | 0.0 |
| **1** | 2500 | 2500 | 2500.0 | 60 months | 15.27 | 59.83 | C | C4 | < 1 year | RENT | ... | INDIVIDUAL | 0.0 |
| **2** | 2400 | 2400 | 2400.0 | 36 months | 15.96 | 84.33 | C | C5 | 10+ years | RENT | ... | INDIVIDUAL | 0.0 |
| **3** | 10000 | 10000 | 10000.0 | 36 months | 13.49 | 339.31 | C | C1 | 10+ years | RENT | ... | INDIVIDUAL | 0.0 |
| **4** | 3000 | 3000 | 3000.0 | 60 months | 12.69 | 67.79 | B | B5 | 1 year | RENT | ... | INDIVIDUAL | 0.0 |

[2] Saar-Tsechansky M. et al. (2007) Handling Missing Values when Applying Classification Models. J. Mach. Learn. Res.

[3] How to Handle Missing Data, https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4

# Data pre-processing

[4]:
```
df.isnull().sum()
```

[4]:
```
loan_amount                 0
funded_amount               0
investor_funds              0
term                        0
interest_rate               0
installment                 0
grade                       0
sub_grade                   0
emp_length              44825
home_ownership              0
annual_income               4
verification_status         0
issue_d                     0
loan_status                 0
pymnt_plan                  0
purpose                     0
addr_state                  0
dti                         0
delinq_2yrs                29
earliest_cr_line           29
inq_last_6mths             29
mths_since_last_delinq 454312
mths_since_last_record 750326
```

[5]:
```
df.emp_length_int.fillna(value=df.emp_length_int.mean(), inplace=True)
df.delinq_2yrs.fillna(value=df.delinq_2yrs.mean(), inplace=True)
df.annual_income.fillna(value=df.annual_income.mean(), inplace=True)
df.open_acc.fillna(value=df.open_acc.mean(), inplace=True)
df.pub_rec.fillna(value=df.pub_rec.mean(), inplace=True)
df.revol_util.fillna(value=df.revol_util.mean(), inplace=True)
df.total_acc.fillna(value=df.total_acc.mean(), inplace=True)
df.collections_12_mths_ex_med.fillna(value=df.collections_12_mths_ex_med.mean(), inplace=True)
df.acc_now_delinq.fillna(value=df.acc_now_delinq.mean(), inplace=True)


# variant of using datetime
# can also be used as time index to calculate any trends
df.next_pymnt_d = pd.to_numeric(df.next_pymnt_d.str.replace('/',''))
df.next_pymnt_d.fillna(value=df.next_pymnt_d.median(), inplace=True)

df.last_credit_pull_d = pd.to_numeric(df.last_credit_pull_d.str.replace('/',''))
df.last_credit_pull_d.fillna(value=df.last_credit_pull_d.median(), inplace=True)

df.final_d = pd.to_numeric(df.final_d.str.replace('/',''))
df.final_d.fillna(value=df.final_d.median(), inplace=True)

df.drop('emp_length', axis=1, inplace=True)
df.drop('earliest_cr_line', axis=1, inplace=True)
df.drop('mths_since_last_delinq', axis=1, inplace=True)
df.drop('mths_since_last_record', axis=1, inplace=True)
df.drop('last_pymnt_d', axis=1, inplace=True)
df.drop('inq_last_6mths', axis=1, inplace=True)
```

[6]:
```
df.isnull().sum()
```

[6]:
```
loan_amount            0
funded_amount          0
```


[2] Saar-Tsechansky M. et al. (2007) Handling Missing Va

[3] How to Handle Missing Data, https://towardsdatascie

# Data pre-processing

## 2b - Encoding

- Manual
- Using LabelEncoder, OrdinalEncoder, OneHotEncoder ...

[9]:
```
df.income_category.unique()
```

[9]:
```
array(['Low', 'Medium', 'High'], dtype=object)
```

[10]:
```
df['income_cat'] = df.income_category.map({'Low':1, 'Medium':2, 'High':3})
df['interest_payment_cat'] = df.interest_payments.map({'Low':1, 'High':2})
df['loan_condition_cat'] = df.loan_condition.map({'Good Loan':0, 'Bad Loan':1})
df['application_type_cat'] = df.application_type.map({'INDIVIDUAL':1, 'JOINT':2})

df['loan_status_cat'] = df.loan_status.map({'Fully Paid':1,
                                             'Charged Off':2,
                                             'Current':3,
                                             'Default':4,
                                             'Late (31-120 days)':5,
                                             'In Grace Period':6,
                                             'Late (16-30 days)':7,
                                             'Does not meet the credit policy. Status:Fully Paid':8,
                                             'Does not meet the credit policy. Status:Charged Off':9,
                                             'Issued':10})
df['verification_status_cat'] = df.verification_status.map({'Verified':1, 'Source Verified':2, 'Not Verified':3})

df['home_ownership_cat'] = df.home_ownership.map({'RENT':1, 'OWN':2, 'MORTGAGE':3, 'OTHER':4, 'NONE':5, 'ANY':6})
df['grade_cat'] = df.grade.map({'A':1, 'B':2, 'C':3, 'D':4, 'E':5, 'F':6, 'G':7})
df['term_cat'] = df.term.map({' 36 months':1, ' 60 months':2})
```

[2] Saar-Tsechansky M. et al. (2007) Handling Missing Values when Applying Classification Models. J. Mach. Learn. Res.

[3] How to Handle Missing Data, https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4

# Data processing

- Visualisation and feature extraction
  - Get insight into the raw data
  - Get insight into the extracted features

- Statistic features
- Trend features
- Semantic features
  - DTI - A debt income ratio - the percentage of a consumer's monthly gross income that goes toward paying debts.

# Data processing

## 3a - Visualisation & Feature extraction

[11]:
```
badloans_df = df.loc[df["loan_condition"] == "Bad Loan"]
goodloans_df = df.loc[df["loan_condition"] == "Good Loan"]
print_string = 'There are {} bad loans and {} good loans in the dataset'.format(badloans_df.shape[0], goodloans_df.shape[0])
print(print_string)
```

```
There are 67429 bad loans and 819950 good loans in the dataset
```

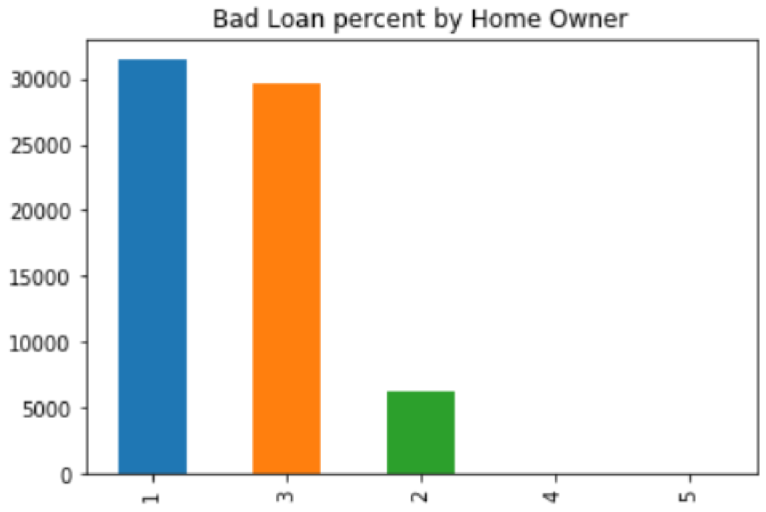
[12]:
```
# loan_status cross
loan_status_cross = pd.crosstab(badloans_df['region'], badloans_df['loan_status']).apply(lambda x: x/x.sum() * 100)
number_of_loanstatus = pd.crosstab(badloans_df['region'], badloans_df['loan_status'])
number_of_loanstatus
```

[12]:

| loan_status | Charged Off | Default | Does not meet the credit policy. Status:Charged Off | In Grace Period | Late (16-30 days) | Late (31-120 days) |
|---|---|---|---|---|---|---|
| **region** | | | | | | |
| **Northern-Irl** | 10671 | 263 | 190 | 1625 | 585 | 2799 |
| **cannught** | 7361 | 175 | 142 | 926 | 354 | 1820 |
| **leinster** | 11094 | 297 | 184 | 1579 | 600 | 2925 |
| **munster** | 4774 | 166 | 79 | 708 | 273 | 1407 |
| **ulster** | 11348 | 318 | 166 | 1415 | 545 | 2640 |

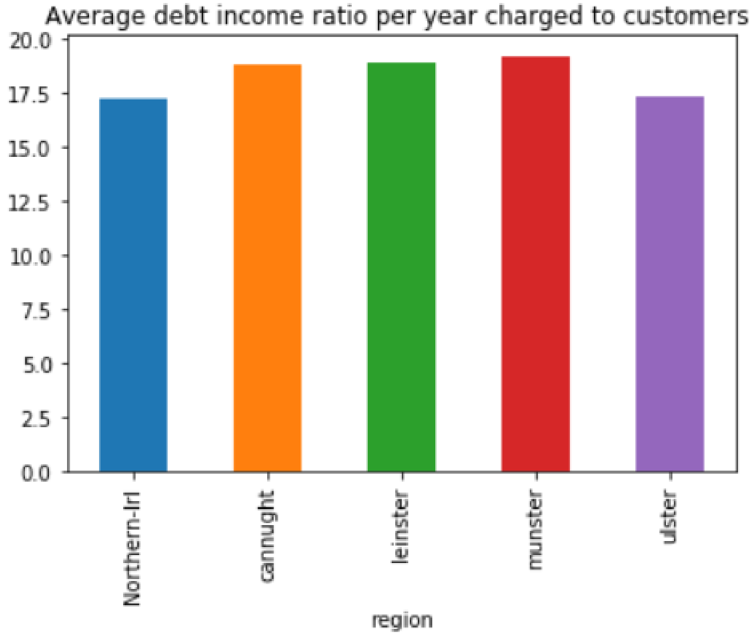
# Data processing

```
[15]: loan_status=df[df.loan_condition_cat== 1].home_ownership_cat.value_counts()
      a = df.home_ownership_cat.unique()
      b = df.home_ownership.unique()
      c = pd.DataFrame(a,b)
      j = loan_status.plot(kind='bar', title='Bad Loan percent by Home Owner')
```




```
[18]: stat4 = df.groupby('region').dti.mean()
      stat4.plot(kind='bar', x='Region', y='debt income ratio ', title=

[18]: <matplotlib.axes._subplots.AxesSubplot at 0x238828052e8>
```

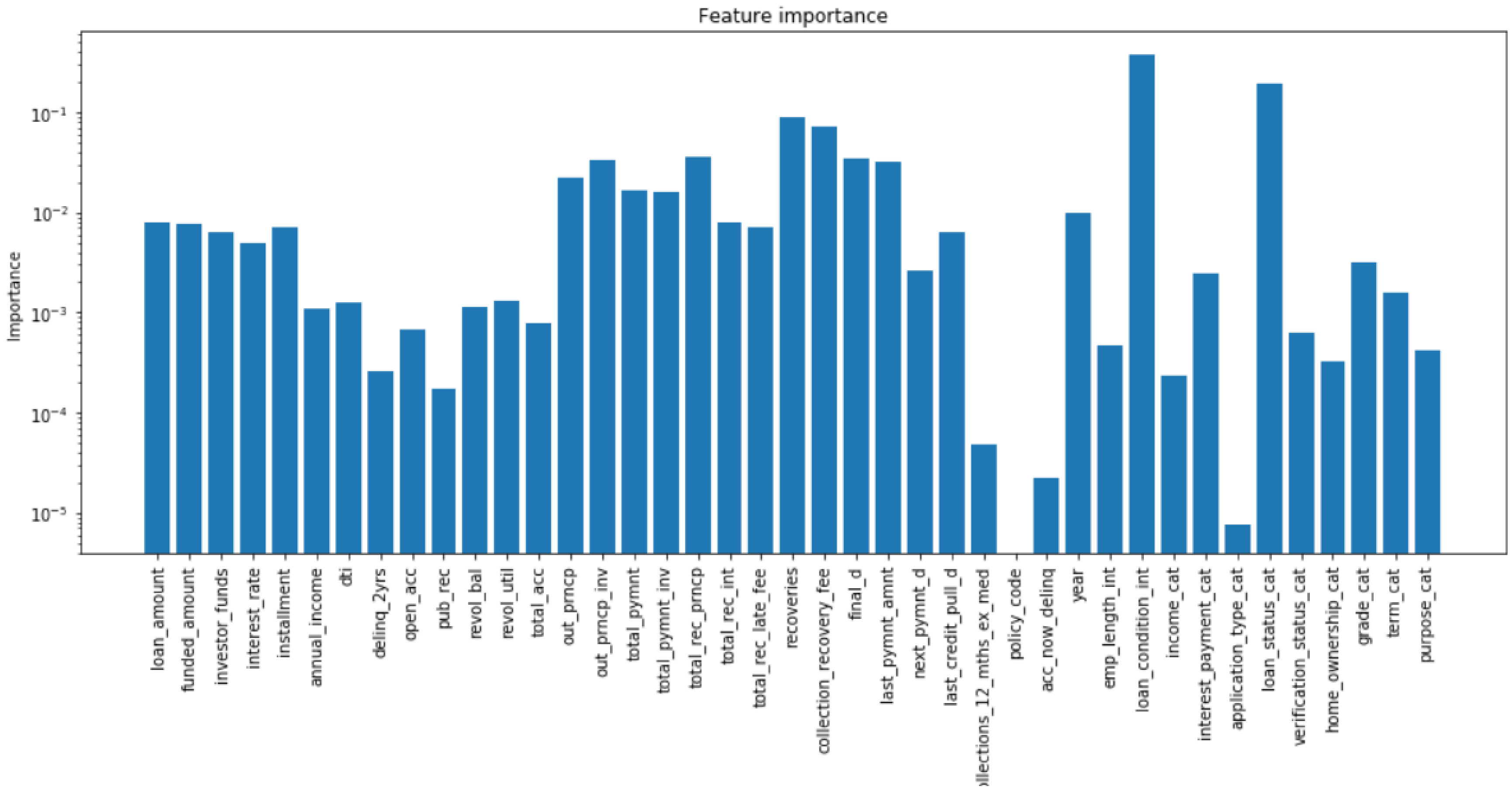


# Training & Evaluation

- **Feature selection** - selection of a subset of relevant features for model training
  - Improves accuracy, reduces training time and reduces overfitting
  1. Rank features
  2. Evaluate subset of features and select the best performing set
- **Training**
  - Experimetn with different ML algorithm for training
  - Analyse the error → ammend the model or models
- **Evaluation**
  - Cross-validation
  - Seperate test dataset

[4] Feature Selection Techniques in Machine Learning with Python, https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e

# Training & Evaluation




[4] Feature Selection Techniques in Machine Learning with Python, https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e

# Training & Evaluation

## 4b - Feature selection

[22]:
```
from sklearn.model_selection import train_test_split
from sklearn import metrics

# procedure that accepts a list of features and evaluates the accuracy score
def train_test_accuracy(feature_cols):
    X = df[feature_cols]
    y = df.loan_condition_cat
    X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=123)
    model = RandomForestClassifier(n_estimators=100, max_features=3, oob_score=True, random_state=1)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    return metrics.accuracy_score(y_test, y_pred)


print (train_test_accuracy(['emp_length_int','annual_income','loan_amount','interest_rate','dti','home_ownership_cat','income_cat','total_pymnt',
```

```
0.9584394509680182
```

[4] Feature Selection Techniques in Machine Learning with Python, https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e

# Training & Evaluation

## 4c Evaluation

- Evaluate different ML models and select the best performing
  - on an seperate test set or
  - in cross-validation

Cross-validation

[23]:
```
feature_cols = ['emp_length_int', 'annual_income','loan_amount',
                'interest_rate','dti','home_ownership_cat',
               'income_cat','total_pymnt','purpose_cat','grade_cat',
               'application_type_cat','term_cat','year']

X = df[feature_cols]
y = df.loan_condition_cat


# random split for cross-validation
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

from sklearn.linear_model import LogisticRegression
Logreg = LogisticRegression()
Logreg.fit(X_train, y_train)

y_pred_class = Logreg.predict(X_test)

from sklearn import metrics
print((metrics.accuracy_score(y_test, y_pred_class))*100)
```

```
92.33158286190809
```

# ML model in use

## VALIDATION

- Real-world imitation
- The trained model is used on yet another dataset
- The accuracy/F-score/error as measured on such dataset can be reported as an objective score

# ML for client behaviour prediction

# The problem - technically

- Train dataset – timeseries per loan (from opening data to closing date if exists)
- For succesfull completition of the task it is required to fuse it with macroeconomical data

| Ime stupca | Tip podatka | Opis |
|---|---|---|
| DATUM_IZVJESTAVANJA | datetime | Datum vremenske serije |
| KLIJENT_ID | numeric | ID |
| OZNAKA_PARTIJE | numeric | Partija |
| DATUM_OTVARANJA | datetime | Datum otvaranja |
| PLANIRANI_DATUM_ZATVARANJA | datetime | Planiran datum zatvaranja |
| DATUM_ZATVARANJA | datetime | Stvarni datum zatvaranja |
| UGOVORENI_IZNOS | numeric | Originalni iznos |
| STANJE_NA_KRAJU_PRETH_KVARTALA | numeric | Preostali iznos kredita na kraju prethodnog kvartala |
| STANJE_NA_KRAJU_KVARTALA | numeric | Preostali iznos kredita na kraju kvartala |
| VALUTA | numeric | Valuta |
| VRSTA_KLIJENTA | numeric | Klijentski segment |
| PROIZVOD | categorical | Produkt |
| VRSTA_PROIZVODA | categorical | Vrsta produkta |
| VISINA_KAMATE | numeric | Iznos visine kamate (postotak) |
| TIP_KAMATE | categorical | Vrsta kamatne stope |
| AGE | numeric | Starost klijenta |
| PRIJEVREMENI_RASKID | categorical | Da/Ne (Y/N) |

# The problem - technically

- Evaluation & Validation dataset – single example per loan
- **It is a prediction task** – you will have to predict whether the client will close the loan before the closing date using only one example describing the loan and the client

| Ime stupca | Tip podatka | Opis |
|---|---|---|
| DATUM_IZVJESTAVANJA | datetime | Datum vremenske serije |
| KLIJENT_ID | numeric | ID |
| OZNAKA_PARTIJE | numeric | Partija |
| DATUM_OTVARANJA | datetime | Datum otvaranja |
| PLANIRANI_DATUM_ZATVARANJA | datetime | Planiran datum zatvaranja |
| UGOVORENI_IZNOS | numeric | Originalni iznos |
| VALUTA | numeric | Valuta |
| VRSTA_KLIJENTA | numeric | Klijentski segment |
| PROIZVOD | categorical | Produkt |
| VRSTA_PROIZVODA | categorical | Vrsta produkta |
| VISINA_KAMATE | numeric | Iznos visine kamate (postotak) |
| TIP_KAMATE | categorical | Vrsta kamatne stope |
| AGE | numeric | Starost klijenta |
| PRIJEVREMENI_RASKID | categorical | YOUR TARGET (Y/N) |

# Requirements

- Documentation & Presentataion
  - Both should follow the template
- Software
  - The software should be structured:
    - Input
    - Data fusion
    - Data preprocessing
    - Data processing
    - Training & Evaluation

| Kriterij | Ocjena | Doprinos ukupnoj ocjeni |
|---|---|---|
| Prezentacija | 0-10 | 10% |
| Dokumentacija | 0-10 | 10% |
| SW – kvaliteta rješenja | 0-35 | 35% |
| Inovativnost rješenja | 0-10 | 10% |
| Ocjena točnosti rješenja<br>Dnevno rangiranje | 0-15 | 15% |
| Validacija rješenja<br>Rangiranje na licu mjesta | 0-20 | 20% |

- Inovation of the solution
  - Whatever you did not hear today and gives you some interesting insights into data
- Evaluation – Daily ranking (web) – at least once
- Validation – On-the-spot ranking – the finalists

# Conclusion

# Conclusions

- Imbalanced dataset
- Try to understand
  - Similarities between loans
  - Similarities between clients
  - The dynamics of payment - the beahviour of the client
- You need to get external data that will help you model the state of macroeconomics during the analysed time
  - From opening to closing the lone
  - How the payment of the loan correlates to country/world economy
- We like visualisations with short descriptions

# Q&A

CONSIDER IT DONE