LAPORAN PROJECT BASED PEMBELAJARAN MESIN

Disusun untuk memenuhi tugas mata kuliah Pembelajaran Mesin

Oleh Kelompok:

1. Herjanto Janawisuta 1301200421 / IF 44 08
2. Alvin Tolopan Armando Sibuea 1301201580 / IF 44 08
3. M Ivan Irsanto 1301200467 / IF 44 08

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

2022

FORMULASI MASALAH

German Credit Risk

Risiko Kredit adalah kemungkinan risiko kerugian akibat kegagalan peminjam untuk membayar kembali pinjaman atau memenuhi kewajiban kontrak. Jika sebuah perusahaan menawarkan kredit kepada kliennya, maka ada risiko bahwa kliennya mungkin tidak membayar tagihan mereka.

Tipe German Credit Risk

Good Risk: Investasi yang diyakini akan menguntungkan. Istilah yang paling sering mengacu pada pinjaman yang diberikan kepada orang atau perusahaan yang layak kredit. Risiko yang baik dianggap sangat mungkin untuk dibayar kembali.

Bad Risk: Pinjaman yang tidak mungkin dilunasi karena sejarah kredit yang buruk, pendapatan yang tidak mencukupi, atau alasan lain. Risiko buruk meningkatkan risiko bagi pemberi pinjaman dan kemungkinan gagal bayar di pihak peminjam.

Objective

Berdasarkan atributnya, mengklasifikasikan seseorang sebagai risiko kredit baik atau buruk.

Deskripsi Dataset

Dataset berisi 1000 columns dengan 20 variabel independen (7 numerik, 13 kategori) dan 1 target variabel. Dalam kumpulan data ini, setiap entri mewakili orang yang mengambil kredit dari bank. Setiap orang diklasifikasikan sebagai risiko kredit baik atau buruk menurut kumpulan atribut.

EKSPLORASI DAN PRA-PEMROSESAN DATA

Import Library dan Pre-Processing

```
import numpy as np
import pandas as pd
import seaborn as sns
from scipy import stats
from math import floor,ceil
import statsmodels.api as sm
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from IPython.display import display, HTML
pd.set_option('display.max_columns', None)
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, confusion_matrix,roc_curve
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.ensemble import RandomForestClassifier
```

Import yang digunakan dalam analisis data German Credit adalah library yang biasa dibutuhkan untuk analisis pre-processing (numpy, pandas, seaborn, matplotlib, sklearn), serta digunakan RandomForestClassifier untuk membuat model bagging dalam proses analisis data. Juga ada beberapa library yang digunakan dalam training dataset (train_test_split, cross_val_value, accuracy_score).

Highlight Cell dalam Columns Dataset

```python
def style_specific_cell(x):

    color_thresh = 'background-color: lightpink'

    df_color = pd.DataFrame('', index=x.index, columns=x.columns)
    rows_number=len(x.index)
    column_number=len(x.columns)
    for r in range(0,rows_number):
        for c in range(0,column_number):
            try:
                val=float(x.iloc[r, c])
                if x.iloc[r, 0]=="Percentage":
                    if val<10:
                        df_color.iloc[r, c]=color_thresh
            except:
                pass

    return df_color

def style_stats_specific_cell(x):

    color_thresh = 'background-color: lightpink'

    df_color = pd.DataFrame('', index=x.index, columns=x.columns)
    rows_number=len(x.index)
    for r in range(0,rows_number):
        try:
            val=(x.iloc[r, 1])
            if val>0.05:
                df_color.iloc[r, 1]=color_thresh
        except:
            pass
    return df_color
```

Program digunakan untuk mentinta (highlight) cell dalam dataset. Dataset yang di highlight adalah Percentage yang digunakan untuk mengetahui persentase credit risk pada beberapa atribut yang dipilih.

Re-Struktur Database berdasarkan Atribut

Jika dilihat dalam Attribute information.docx, terdapat beberapa nilai atribut yang diklasifikasi berdasarkan tipe atribut (seperti A71 = unemployment), untuk itu seluruh dataset perlu di struktur ulang berdasarkan nilai atribut dataset yang sesuai.

Program akan menstruktur ulang seperti berikut:

```
#Memasukkan database ke colab
df= pd.read_csv("http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data",sep=" ",header=None)
headers=["Status of existing checking account","Duration in month","Credit history",
         "Purpose","Credit amount","Savings account/bonds","Present employment since",
         "Installment rate in percentage of disposable income","Personal status and sex",
         "Other debtors / guarantors","Present residence since","Property","Age in years",
        "Other installment plans","Housing","Number of existing credits at this bank",
        "Job","Number of people being liable to provide maintenance for","Telephone","foreign worker","Cost Matrix(Risk)"]
df.columns= headers
df.to_csv("german_data_credit_cat.csv",index=False)
```

```
#Menstrukturkan database berdasarkan attribute
Status_of_existing_checking_account={'A14':"no checking account",'A11':"<0 DM", 'A12': "0 <= <200 DM",'A13':">= 200 DM "}
df["Status of existing checking account"]=df["Status of existing checking account"].map(Status_of_existing_checking_account)

Credit_history={"A34":"critical account","A33":"delay in paying off","A32":"existing credits paid back duly till now","A31":"all credits at this bank paid back duly","A30":"no credits taken"}
df["Credit history"]=df["Credit history"].map(Credit_history)

Purpose={"A40" : "car (new)", "A41" : "car (used)", "A42" : "furniture/equipment", "A43" :"radio/television" , "A44" : "domestic appliances", "A45" : "repairs", "A46" : "education",
        'A47' : 'vacation','A48' : 'retraining','A49' : 'business','A410' : 'others'}
df["Purpose"]=df["Purpose"].map(Purpose)

Saving_account={"A65" : "no savings account","A61" :"<100 DM","A62" : "100 <= <500 DM","A63" :"500 <= < 1000 DM", "A64" :">= 1000 DM"}
df["Savings account/bonds"]=df["Savings account/bonds"].map(Saving_account)

Present_employment={'A75':">=7 years", 'A74':"4<= <7 years",  'A73':"1<= < 4 years", 'A72':"<1 years",'A71':"unemployed"}
df["Present employment since"]=df["Present employment since"].map(Present_employment)

Personal_status_and_sex={ 'A95':"female:single",'A94':"male:married/widowed",'A93':"male:single", 'A92':"female:divorced/separated/married", 'A91':"male:divorced/separated"}
df["Personal status and sex"]=df["Personal status and sex"].map(Personal_status_and_sex)
```

```
Other_debtors_guarantors={'A101':"none", 'A102':"co-applicant", 'A103':"guarantor"}
df["Other debtors / guarantors"]=df["Other debtors / guarantors"].map(Other_debtors_guarantors)

Property={'A121':"real estate", 'A122':"savings agreement/life insurance", 'A123':"car or other", 'A124':"unknown / no property"}
df["Property"]=df["Property"].map(Property)

Other_installment_plans={'A143':"none", 'A142':"store", 'A141':"bank"}
df["Other installment plans"]=df["Other installment plans"].map(Other_installment_plans)

Housing={'A153':"for free", 'A152':"own", 'A151':"rent"}
df["Housing"]=df["Housing"].map(Housing)

Job={'A174':"management/ highly qualified employee", 'A173':"skilled employee / official", 'A172':"unskilled - resident", 'A171':"unemployed/ unskilled  - non-resident"}
df["Job"]=df["Job"].map(Job)

Telephone={'A192':"yes", 'A191':"none"}
df["Telephone"]=df["Telephone"].map(Telephone)

foreign_worker={'A201':"yes", 'A202':"no"}
df["foreign worker"]=df["foreign worker"].map(foreign_worker)

risk={1:"Good Risk", 2:"Bad Risk"}
df["Cost Matrix(Risk)"]=df["Cost Matrix(Risk)"].map(risk)
```

Hasil Dataframe setelah Di Re-Struktur adalah:

```
df.sample(5)
```

	Status of existing checking account	Duration in month	Credit history	Purpose	Credit amount	Savings account/bonds	Present employment since	Installment rate in percentage of disposable income	Personal status and sex	Other debtors / guarantors	Present residence since	Property	Age in years	Other installment plans	Housing
186	0 <= <200 DM	9	all credits at this bank paid back duly	car (used)	5129	<100 DM	>=7 years	2	female:divorced/separated/married	none	4	unknown / no property	74	bank	for free
184	0 <= <200 DM	18	critical account	car (new)	884	<100 DM	>=7 years	4	male:single	none	4	car or other	36	bank	own
189	0 <= <200 DM	18	no credits taken	furniture/equipment	3244	<100 DM	1<= < 4 years	1	female:divorced/separated/married	none	4	car or other	33	bank	own
510	<0 DM	12	existing credits paid back duly till now	car (new)	759	<100 DM	4<= <7 years	4	male:single	none	2	real estate	26	none	own
85	no checking account	12	critical account	business	1412	<100 DM	1<= < 4 years	4	female:divorced/separated/married	guarantor	2	real estate	29	none	own

Info tipe dataset per attribut:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 21 columns):
 #   Column                                                    Non-Null Count  Dtype 
---  ------                                                    --------------  ----- 
 0   Status of existing checking account                       1000 non-null   object
 1   Duration in month                                         1000 non-null   int64 
 2   Credit history                                            1000 non-null   object
 3   Purpose                                                   1000 non-null   object
 4   Credit amount                                             1000 non-null   int64 
 5   Savings account/bonds                                     1000 non-null   object
 6   Present employment since                                  1000 non-null   object
 7   Installment rate in percentage of disposable income       1000 non-null   int64 
 8   Personal status and sex                                   1000 non-null   object
 9   Other debtors / guarantors                                1000 non-null   object
 10  Present residence since                                   1000 non-null   int64 
 11  Property                                                  1000 non-null   object
 12  Age in years                                              1000 non-null   int64 
 13  Other installment plans                                   1000 non-null   object
 14  Housing                                                   1000 non-null   object
 15  Number of existing credits at this bank                   1000 non-null   int64 
 16  Job                                                       1000 non-null   object
 17  Number of people being liable to provide maintenance for  1000 non-null   int64 
 18  Telephone                                                 1000 non-null   object
 19  foreign worker                                            1000 non-null   object
 20  Cost Matrix(Risk)                                         1000 non-null   object
dtypes: int64(7), object(14)
memory usage: 164.2+ KB
```

Summary Persentase Setiap Kategori

Program akan memberi persentase pada setiap kategori pada tiap atribut. Program akan meng highlight seluruh persentase yang memiliki bad risk pada setiap kategori.

```
#menghapuskan (drop) columns yang merupakan numerical variable
column_names=df.columns.tolist()
column_names.remove("Credit amount") #numerical variable
column_names.remove("Age in years") #numerical variable
column_names.remove("Duration in month") #numerical variable

column_names_cat={}
for name in column_names:
    column_names_cat[name]=len(df[name].unique().tolist())

    marginal_report_cluster={}
for itr in range(0,np.asarray(list(column_names_cat.values())).max()+1):
    if [k for k,v in column_names_cat.items() if v == itr]:
        marginal_report_cluster[itr]=[k for k,v in column_names_cat.items() if v == itr]
```

```
#Memberikan dan meng-highlight persentase risk pada credit (dalam tabel) berserta attribut
for key in marginal_report_cluster.keys():
    marginal_percentage_report=[]
    for name in sorted(marginal_report_cluster[key]):
        data=pd.crosstab(df[name],columns=["Percentage"]).apply(lambda r: (round((r/r.sum())*100,2)), axis=0).reset_index()
        data.columns=[name,"Percentage"]
        data=data.transpose().reset_index()
        [marginal_percentage_report.append(x) for x in data.values.tolist()]
        options=[]
    marginal_percentage_report=pd.DataFrame(marginal_percentage_report)
    [options.append("Category Option "+str(itr)) for itr in range(1,len(marginal_percentage_report.columns))]
    marginal_percentage_report.columns=["Attribute"]+options
    display(marginal_percentage_report.style.apply(style_specific_cell, axis=None))
```

Hasil:

	Attribute	Category Option 1	Category Option 2
0	Cost Matrix(Risk)	Bad Risk	Good Risk
1	Percentage	30.000000	70.000000
2	Number of people being liable to provide maintenance for	1.000000	2.000000
3	Percentage	84.500000	15.500000
4	Telephone	none	yes
5	Percentage	59.600000	40.400000
6	foreign worker	no	yes
7	Percentage	3.700000	96.300000

	Attribute	Category Option 1	Category Option 2	Category Option 3
0	Housing	for free	own	rent
1	Percentage	10.800000	71.300000	17.900000
2	Other debtors / guarantors	co-applicant	guarantor	none
3	Percentage	4.100000	5.200000	90.700000
4	Other installment plans	bank	none	store
5	Percentage	13.900000	81.400000	4.700000

	Attribute	Category Option 1	Category Option 2	Category Option 3	Category Option 4
0	Installment rate in percentage of disposable income	1.000000	2.000000	3.000000	4.000000
1	Percentage	13.600000	23.100000	15.700000	47.600000
2	Job	management/ highly qualified employee	skilled employee / official	unemployed/ unskilled - non-resident	unskilled - resident
3	Percentage	14.800000	63.000000	2.200000	20.000000
4	Number of existing credits at this bank	1.000000	2.000000	3.000000	4.000000
5	Percentage	63.300000	33.300000	2.800000	0.600000
6	Personal status and sex	female:divorced/separated/married	male:divorced/separated	male:married/widowed	male:single
7	Percentage	31.000000	5.000000	9.200000	54.800000
8	Present residence since	1.000000	2.000000	3.000000	4.000000
9	Percentage	13.000000	30.800000	14.900000	41.300000
10	Property	car or other	real estate	savings agreement/life insurance	unknown / no property
11	Percentage	33.200000	28.200000	23.200000	15.400000
12	Status of existing checking account	0 <= <200 DM	<0 DM	>= 200 DM	no checking account
13	Percentage	26.900000	27.400000	6.300000	39.400000

	Attribute	Category Option 1	Category Option 2	Category Option 3	Category Option 4	Category Option 5
0	Credit history	all credits at this bank paid back duly	critical account	delay in paying off	existing credits paid back duly till now	no credits taken
1	Percentage	4.900000	29.300000	8.800000	53.000000	4.000000
2	Present employment since	1<= < 4 years	4<= <7 years	<1 years	>=7 years	unemployed
3	Percentage	33.900000	17.400000	17.200000	25.300000	6.200000
4	Savings account/bonds	100 <= <500 DM	500 <= < 1000 DM	<100 DM	>= 1000 DM	no savings account
5	Percentage	10.300000	6.300000	60.300000	4.800000	18.300000

	Attribute	Category Option 1	Category Option 2	Category Option 3	Category Option 4	Category Option 5	Category Option 6	Category Option 7	Category Option 8	Category Option 9	Category Option 10
0	Purpose	business	car (new)	car (used)	domestic appliances	education	furniture/equipment	others	radio/television	repairs	retraining
1	Percentage	9.700000	23.400000	10.300000	1.200000	5.000000	18.100000	1.200000	28.000000	2.200000	0.900000

Menggabungkan Atribut

```
df=pd.read_csv("german_data_credit_cat.csv")
number_of_credit={1:1,2:2,3:2,4:2}
df["Number of existing credits at this bank"]=df["Number of existing credits at this bank"].map(number_of_credit)


Status_of_existing_checking_account={'A14':"no checking account",'A11':"<0 DM", 'A12': ">0 DM",'A13':">0 DM"}
df["Status of existing checking account"]=df["Status of existing checking account"].map(Status_of_existing_checking_account)
```

```
Credit_history={"A34":"critical account/delay in paying off","A33":"critical account/delay in paying off","A32":"all credit / existing credits paid back duly till now","A31":"all credit / exist
df["Credit history"]=df["Credit history"].map(Credit_history)
Purpose={"A40" : "car (new)", "A41" : "car (used)", "A42" : "Home Related", "A43" :"Home Related" , "A44" : "Home Related", "A45" : "Home Related", "A46" : "others", 'A47' : 'others','A48' : 'o
df["Purpose"]=df["Purpose"].map(Purpose)


Saving_account={"A65" : "no savings account","A61" :"<100 DM","A62" : "<500 DM","A63" :">500 DM", "A64" :">500 DM"}
df["Savings account/bonds"]=df["Savings account/bonds"].map(Saving_account)


Present_employment={'A75':">=7 years", 'A74':"4<= <7 years",  'A73':"1<= < 4 years", 'A72':"<1 years",'A71':"<1 years"}
df["Present employment since"]=df["Present employment since"].map(Present_employment)
```

```
Personal_status_and_sex={ 'A95':"female",'A94':"male",'A93':"male", 'A92':"female", 'A91':"male"}
df["Personal status and sex"]=df["Personal status and sex"].map(Personal_status_and_sex)


Other_debtors_guarantors={'A101':"none", 'A102':"co-applicant/guarantor", 'A103':"co-applicant/guarantor"}
df["Other debtors / guarantors"]=df["Other debtors / guarantors"].map(Other_debtors_guarantors)

Property={'A121':"real estate", 'A122':"savings agreement/life insurance", 'A123':"car or other", 'A124':"unknown / no property"}
df["Property"]=df["Property"].map(Property)

Other_installment_plans={'A143':"none", 'A142':"bank/store", 'A141':"bank/store"}
df["Other installment plans"]=df["Other installment plans"].map(Other_installment_plans)

Housing={'A153':"for free", 'A152':"own", 'A151':"rent"}
df["Housing"]=df["Housing"].map(Housing)

Job={'A174':"employed", 'A173':"employed", 'A172':"unemployed", 'A171':"unemployed"}
df["Job"]=df["Job"].map(Job)

Telephone={'A192':"yes", 'A191':"none"}
df["Telephone"]=df["Telephone"].map(Telephone)

foreign_worker={'A201':"yes", 'A202':"no"}
df["foreign worker"]=df["foreign worker"].map(foreign_worker)

risk={1:"Good Risk", 2:"Bad Risk"}
df["Cost Matrix(Risk)"]=df["Cost Matrix(Risk)"].map(risk)
```

Tampilan dataframe setelah digabung

	Status of existing checking account	Duration in month	Credit history	Purpose	Credit amount	Savings account/bonds	Present employment since	Installment rate in percentage of disposable income	Personal status and sex	Other debtors / guarantors	Present residence since	Property	Age in years	Other installment plans	Housing
0	<0 DM	6	critical account/delay in paying off	Home Related	1169	no savings account	>=7 years	4	male	none	4	real estate	67	none	own
1	>0 DM	48	all credit / existing credits paid back duly t...	Home Related	5951	<100 DM	1<= < 4 years	2	female	none	2	real estate	22	none	own
2	no checking account	12	critical account/delay in paying off	others	2096	<100 DM	4<= <7 years	2	male	none	3	real estate	49	none	own

Number of existing credits at this bank	Job	Number of people being liable to provide maintenance for	Telephone	foreign worker	Cost Matrix(Risk)
2	employed	1	yes	yes	Good Risk
1	employed	1	none	yes	Bad Risk
1	unemployed	2	none	yes	Good Risk

Dataframe akan dieksplorasi lagi seperti diatas untuk memperbagus hasil pre-processing.

Melihatkan Numerical Variable

```
df[["Credit amount","Age in years","Duration in month"]].describe()
```

	Credit amount	Age in years	Duration in month
count	1000.000000	1000.000000	1000.000000
mean	3271.258000	35.546000	20.903000
std	2822.736876	11.375469	12.058814
min	250.000000	19.000000	4.000000
25%	1365.500000	27.000000	12.000000
50%	2319.500000	33.000000	18.000000
75%	3972.250000	42.000000	24.000000
max	18424.000000	75.000000	72.000000

Eksplorasi Data

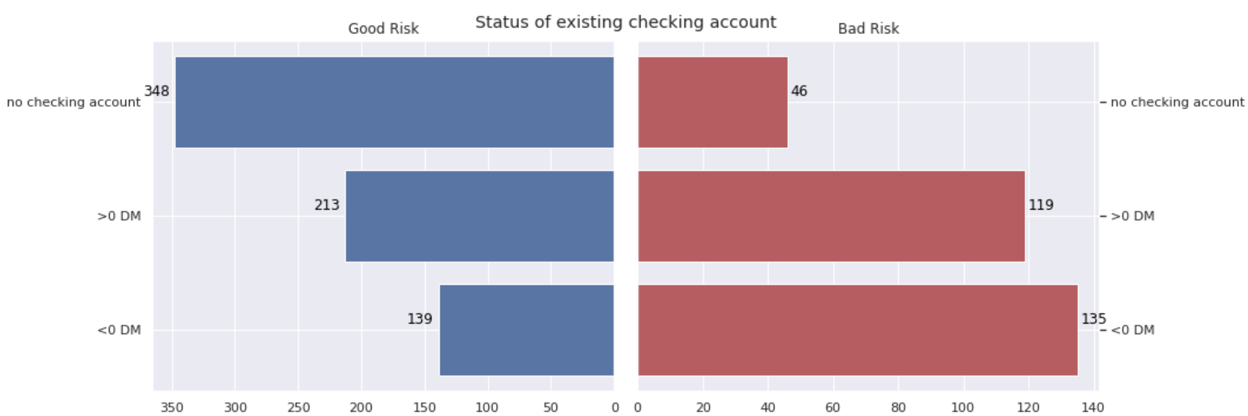
Function Pembuatan Barplot

```
def visualize_distribution(attr):
    good_risk_df = df[df["Cost Matrix(Risk)"]=="Good Risk"]
    bad_risk_df = df[df["Cost Matrix(Risk)"]=="Bad Risk"]
    fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(15,5))
    attr_good_risk_df = good_risk_df[[attr, 'Cost Matrix(Risk)']].groupby(attr).count()
    attr_bad_risk_df = bad_risk_df[[attr, 'Cost Matrix(Risk)']].groupby(attr).count()
    ax[0].barh( attr_good_risk_df['Cost Matrix(Risk)'].index.tolist(), attr_good_risk_df['Cost Matrix(Risk)'].tolist(), align='center', color="#5975A4")
    ax[1].barh( attr_bad_risk_df['Cost Matrix(Risk)'].index.tolist(), attr_bad_risk_df['Cost Matrix(Risk)'].tolist(), align='center', color="#B55D60")
    ax[0].set_title('Good Risk')
    ax[1].set_title('Bad Risk')
    ax[0].invert_xaxis()
    ax[1].yaxis.tick_right()

    num_para_change=["Present residence since","Number of existing credits at this bank","Installment rate in percentage of disposable income","Number of people being liable to provide maintenar
    if attr in num_para_change:
        for i, v in enumerate(attr_good_risk_df['Cost Matrix(Risk)'].tolist()):
            ax[0].text(v+15, i+1, str(v), color='black')
        for i, v in enumerate(attr_bad_risk_df['Cost Matrix(Risk)'].tolist()):
            ax[1].text(v+2, i+1, str(v), color='black')
    else:
        for i, v in enumerate(attr_good_risk_df['Cost Matrix(Risk)'].tolist()):
            ax[0].text(v+25, i + .05, str(v), color='black')
        for i, v in enumerate(attr_bad_risk_df['Cost Matrix(Risk)'].tolist()):
            ax[1].text(v+1, i + .05, str(v), color='black')
    plt.suptitle(attr)
    plt.tight_layout()
    plt.show()
```

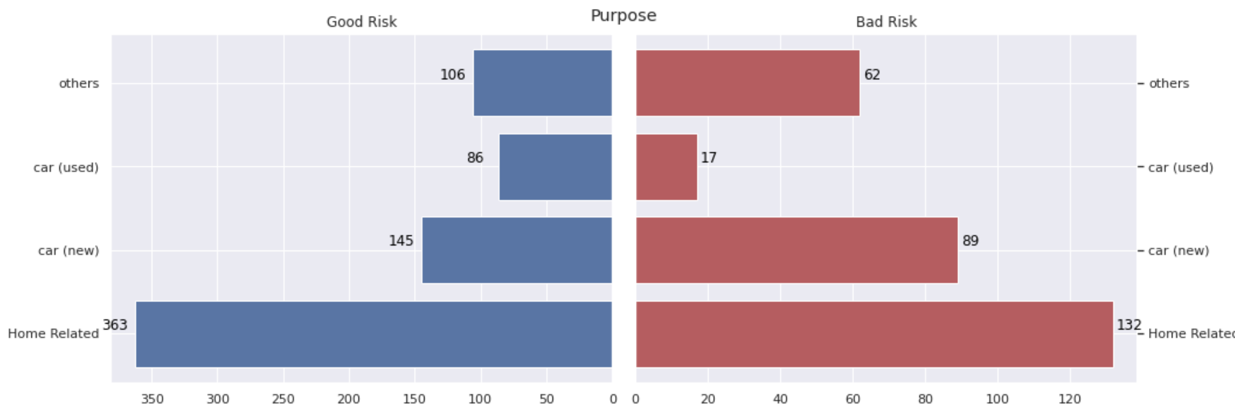
Visualisasi Data

Status of existing checking account

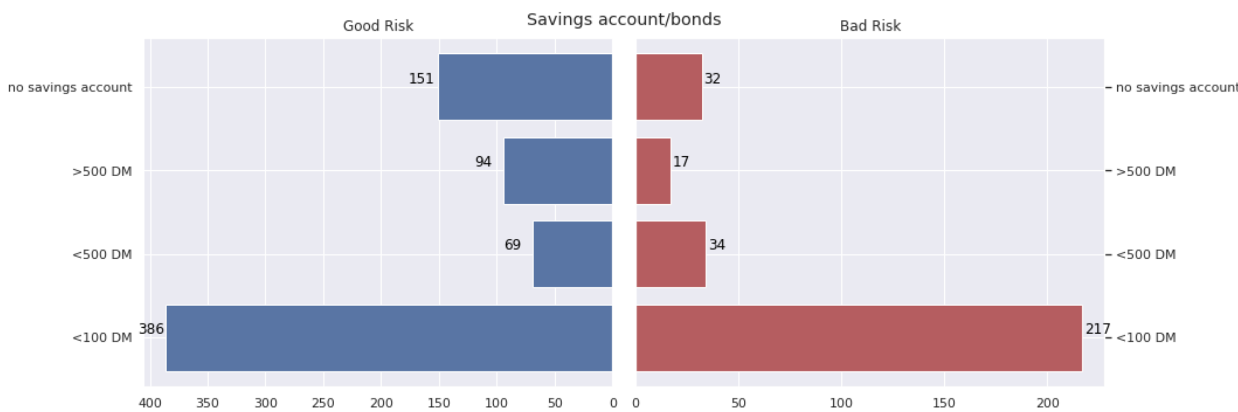


Credit History

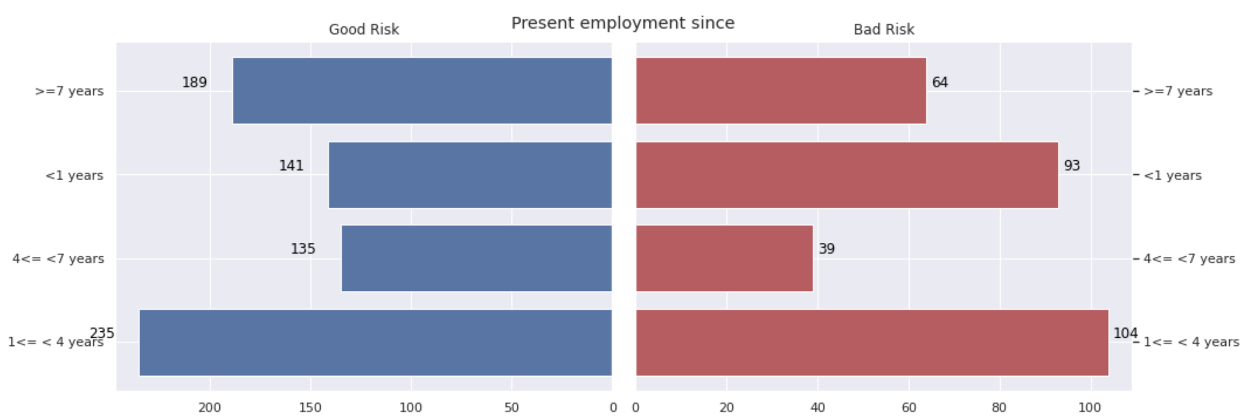
Purpose



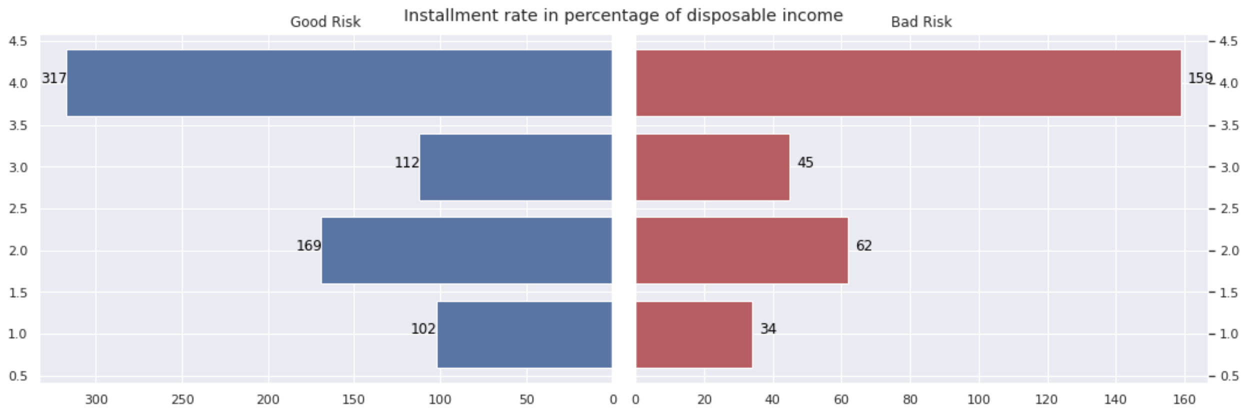
Savings account/bonds



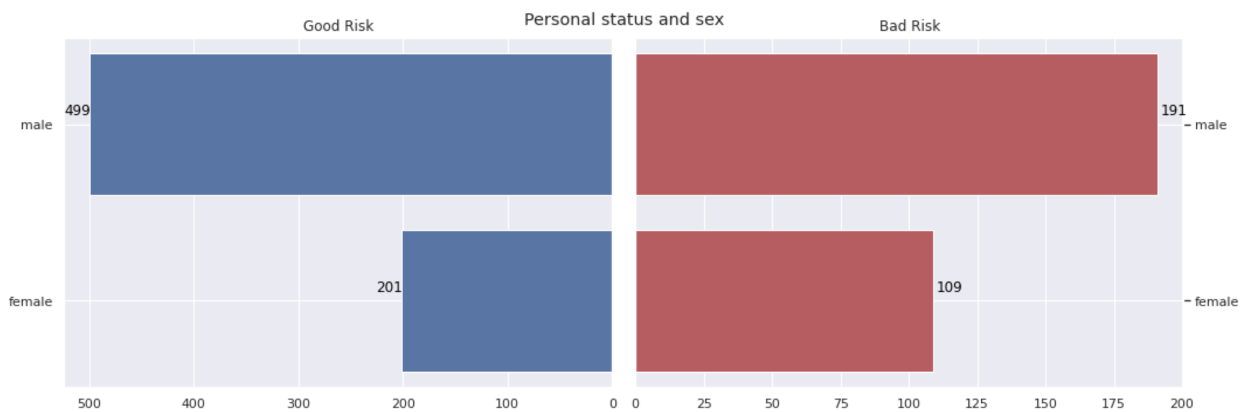
Present employment since



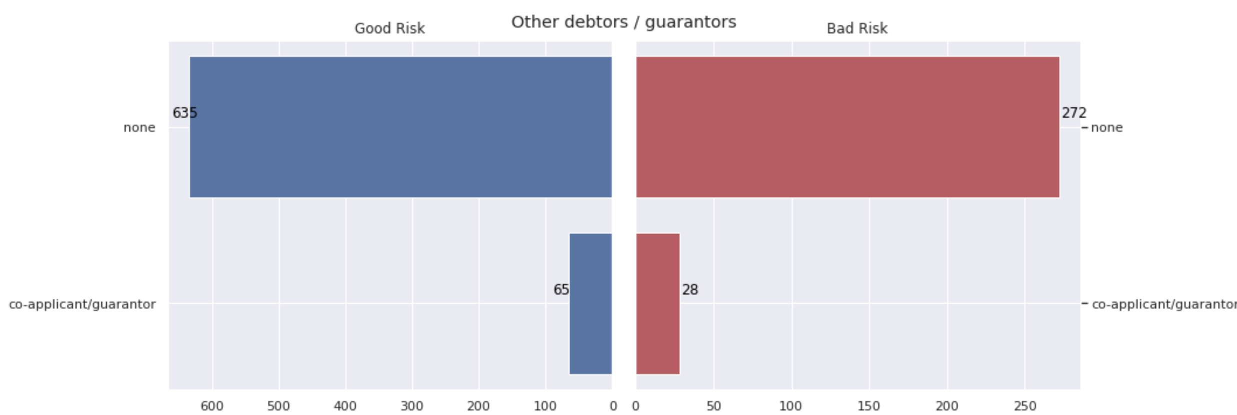
Installment rate in percentage of disposable income



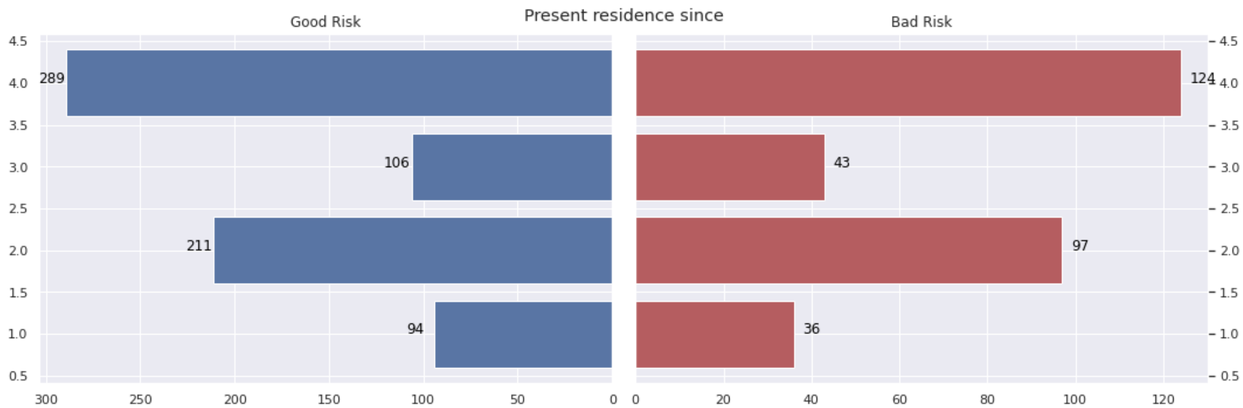
Personal status and sex



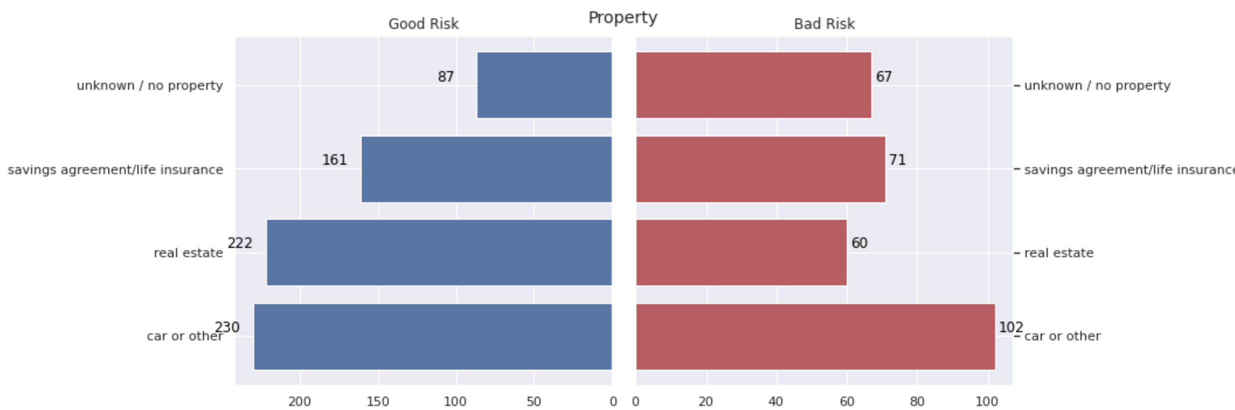
Other deptors / guarantors



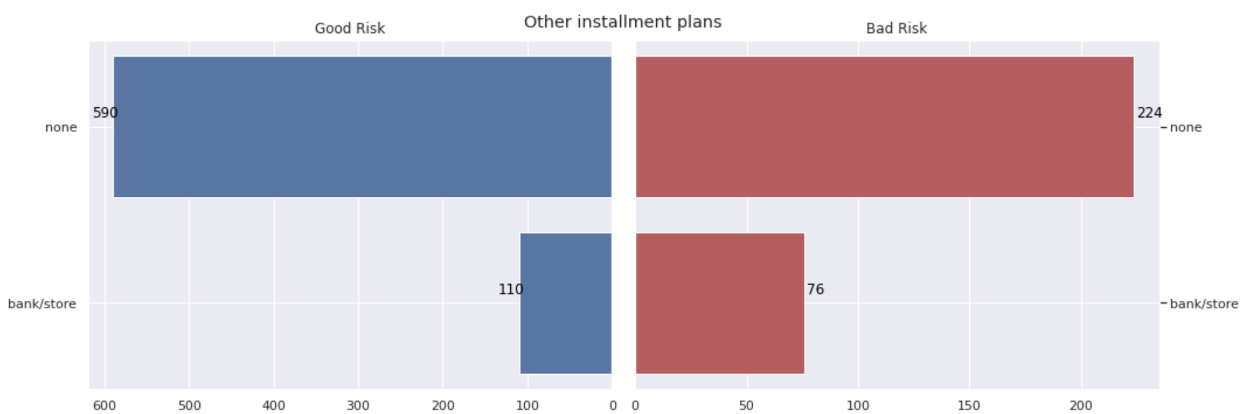
Present residence since



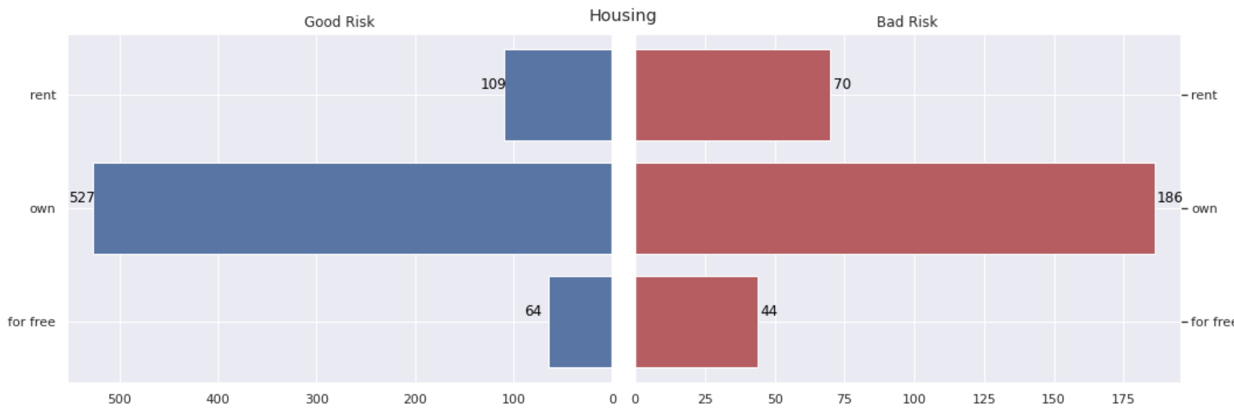
Property



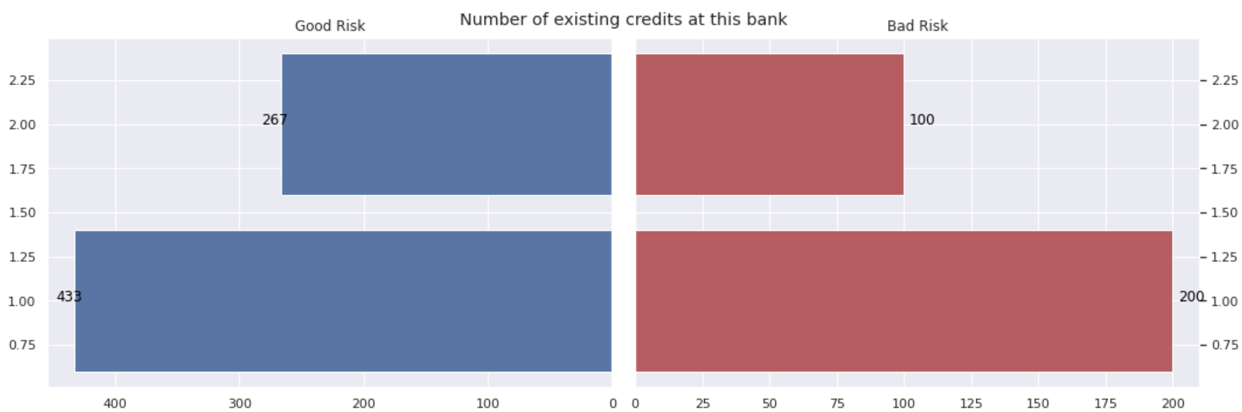
Other installment plans



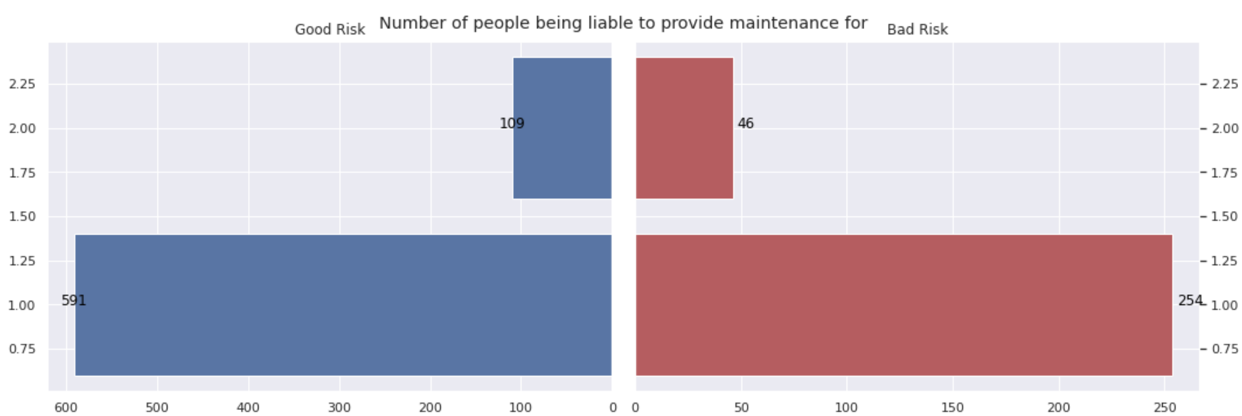
Housing



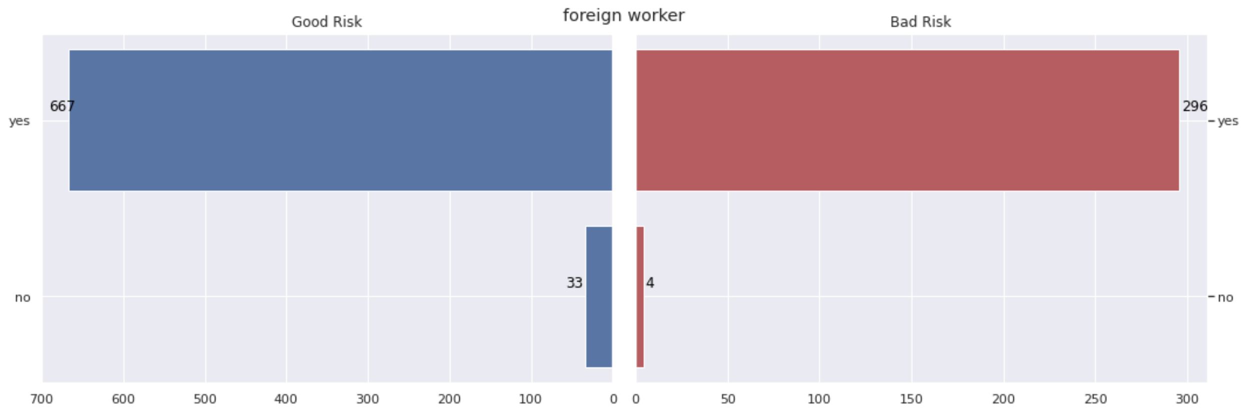
Number of existing credits at this bank



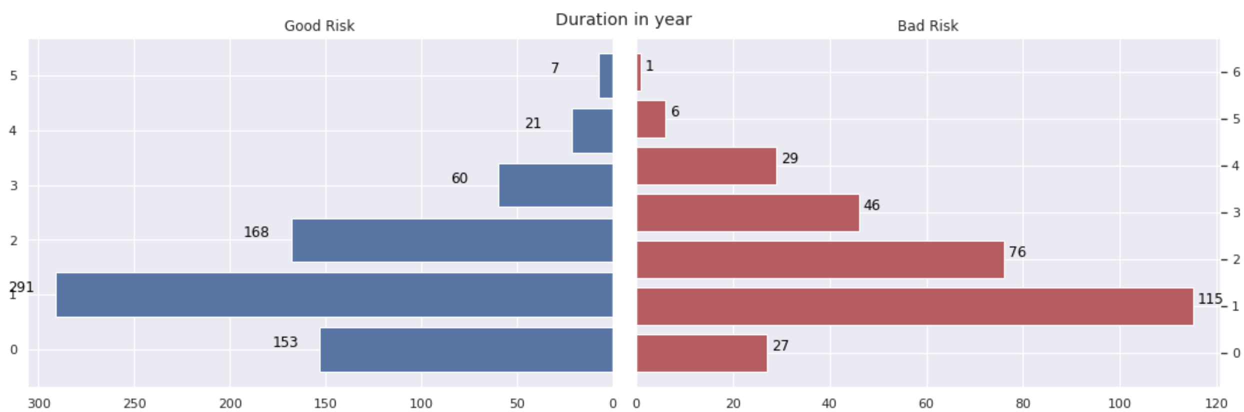
Number of people being liable to provide maintenance for



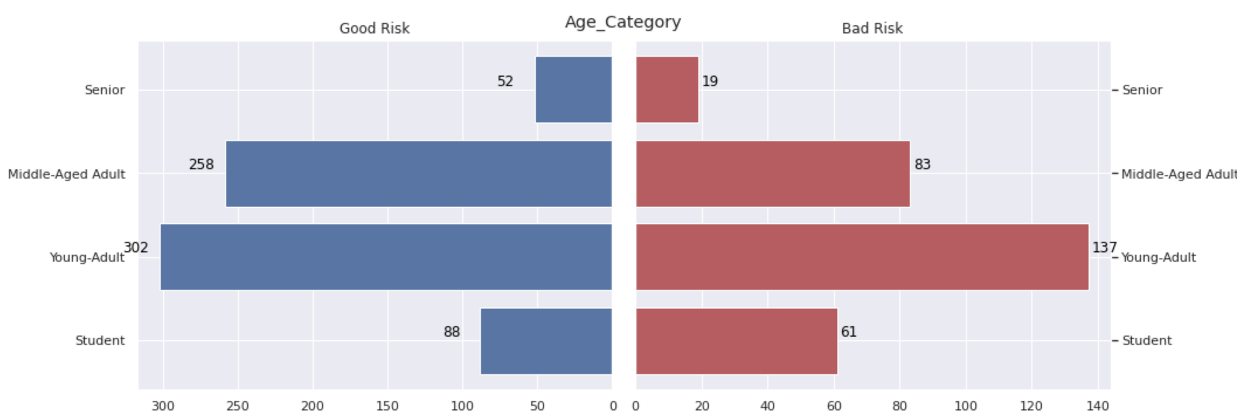
Foreign worker



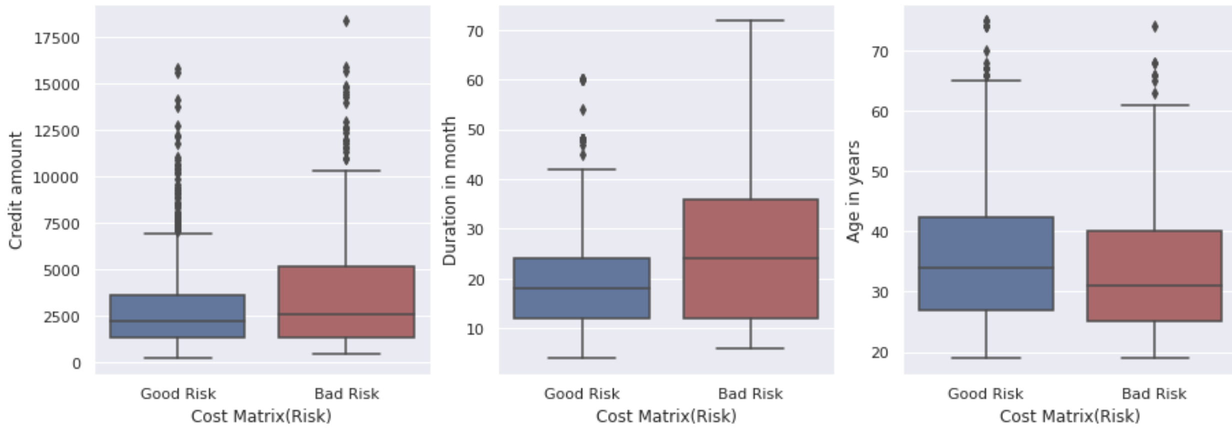
Duration in year



Age category



Boxplot Cost Matrix berdasarkan Numerical Variable



Feature Selection

	Attribute	P-value
0	Status of existing checking account	0.000000
1	Credit history	0.000000
2	Purpose	0.000050
3	Savings account/bonds	0.000000
4	Present employment since	0.000422
5	Installment rate in percentage of disposable income	0.140033
6	Personal status and sex	0.020740
7	Other debtors / guarantors	1.000000
8	Present residence since	0.861552
9	Property	0.000029
10	Other installment plans	0.000476
11	Housing	0.000112
12	Number of existing credits at this bank	0.169304
13	Job	0.606737
14	Number of people being liable to provide maintenance for	1.000000
15	Telephone	0.278876
16	foreign worker	0.015831

	Attribute	P-value
0	Credit amount	0.000001
1	Age in years	0.003925
2	Duration in month	0.000000

PEMODELAN

Untuk Pemodelan, kita akan menggunakan 12 attribute significant dengan 1 target variable untuk membuat model. Seluruh atribut tersebut akan digabung, Serta column risk akan diganti menjadi numeric.

```
attr_significant=["Status of existing checking account","Credit history","Purpose",\
"Savings account/bonds","Present employment since",\
"Personal status and sex","Property","Other installment plans","Housing","foreign worker",\
"Credit amount","Age in years","Duration in month"]
target_variable=["Cost Matrix(Risk)"]
df=df[attr_significant+target_variable]
```

```
[262] col_cat_names=["Status of existing checking account","Credit history","Purpose",\
"Savings account/bonds","Present employment since",\
"Personal status and sex","Property","Other installment plans","Housing","foreign worker"]
for attr in col_cat_names:
    df = df.merge(pd.get_dummies(df[attr], prefix=attr), left_index=True, right_index=True)
    df.drop(attr,axis=1,inplace=True)

#mengubah target variable menjadi numeric
risk={"Good Risk":1, "Bad Risk":0}
df["Cost Matrix(Risk)"]=df["Cost Matrix(Risk)"].map(risk)
```

Pemodelan kita akan menggunakan test PCA serta test menggunakan Bagging. Kita akan memasukkan 16 components dan model akan di train (dengan training set di split, menggunakan test size sebesar 0.30 tanpa menggunakan random_state). Model akan menggunakan RandomForestClassifier untuk bekerja.

```
X = df.drop('Cost Matrix(Risk)', 1).values #independent variables
y = df["Cost Matrix(Risk)"].values #target variables

pca = PCA(n_components=16)
X = pca.fit_transform(X)
```

```
[266] model=RandomForestClassifier()
```

Dataset akan dikategorikan menjadi training set X dan Y, serta testing set X dan y.

```
[265] # Spliting dataset into train and test version
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30,random_state=0)
```

EVALUASI

Hasil menggunakan PCA akan menghasilkan akurasi data sebesar 76%

```
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy: ")
print(round(accuracy_score(y_test,y_pred)*100,2))
```

```
Accuracy:
76.0
```

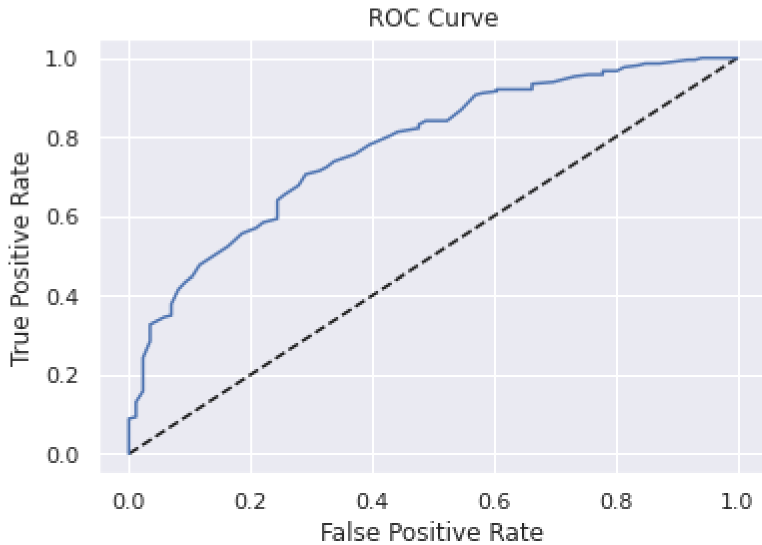
Dengan menggunakan bagging classifier, akurasi data akan mengasilkan 0.7467% atau 74.67%

```
[269] from sklearn.ensemble import BaggingClassifier
      base_classifier = RandomForestClassifier(n_estimators=10)
      bagging_classifier = BaggingClassifier(base_estimator=base_classifier,
                                             n_estimators=10,
                                             max_samples=0.8,
                                             max_features=0.8)
      # Train the classifier
      bagging_classifier.fit(X_train, y_train)

      # Test the classifier
      bagging_classifier.score(X_test, y_test)
```

```
0.7466666666666667
```

Kami juga menggunakan ROC curve untuk mengetahui hasil berdasarkan kurva. Dari hasil ROC dinyatakan bahwa ROC Curve bisa dievaluasi bahwa hasil nilai mendekati ke 0.7 (mendekati ke 1), yang artinya model yang dibuat memiliki >70% area dibawah kurva.



EKSPERIMEN

Untuk melakukan eksperimen, kita akan mengubah test_size menjadi 0.15 untuk mengetahui hasil akurasi data jika test_size dikurangi.

```
[282] # Spliting dataset into train and test version
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.15,random_state=0)
```

Hasil

Dengan menggunakan PCA dengan Model RandomForestClassifier, akurasi data mencapai 77.33%

```
[284] model=RandomForestClassifier()

[285] model.fit(X_train, y_train)
      y_pred = model.predict(X_test)
      print("Accuracy: ")
      print(round(accuracy_score(y_test,y_pred)*100,2))
```

```
Accuracy:
77.33
```

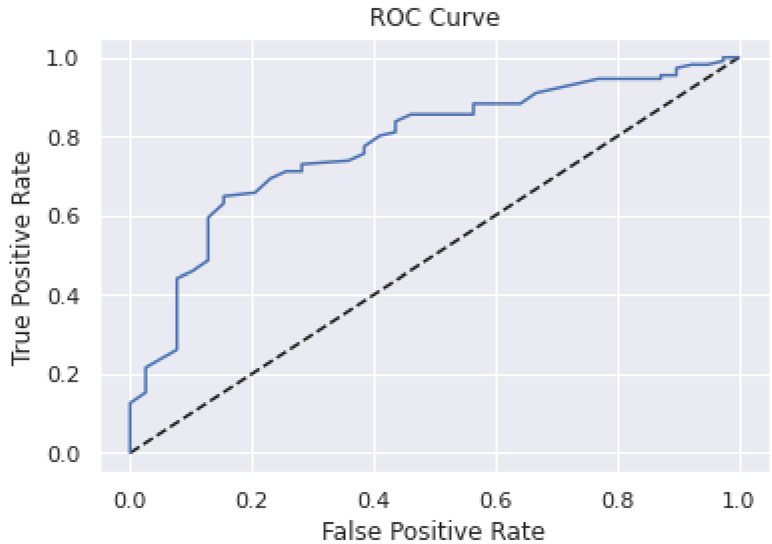
Sedangkan menggunakan Bagging Classifier hasil data akan menghasilkan akurasi data sebesar 0.76% atau 76%

```
from sklearn.ensemble import BaggingClassifier
base_classifier = RandomForestClassifier(n_estimators=10)
bagging_classifier = BaggingClassifier(base_estimator=base_classifier,
                                       n_estimators=10,
                                       max_samples=0.8,
                                       max_features=0.8)
# Train the classifier
bagging_classifier.fit(X_train, y_train)

# Test the classifier
bagging_classifier.score(X_test, y_test)
```

```
0.76
```

Ditambah dengan Menggunakan ROC Curve, disimpulkan bahwa Dari hasil eksperimen, bisa disimpulkan bahwa akurasi data semakin membaik jika test_size dikurangi.



KESIMPULAN

Berdasarkan Hasil dari Dataset German Credit, bisa disimpulkan bahwa hasil akurasi data bisa berubah berdasarkan test_size yang diisi dan jenis model algoritma yang digunakan. Dengan tiap perbedaan model terdapat perbedaan 0.7-1% dan perbedaan test_size terdapat perbedaan akurasi sebesar 0.6-1%

LAMPIRAN

Link Google Colab:

https://colab.research.google.com/drive/109B3dwasMg9k6bKS9v4_egcm9ipY-Ra1#scrollTo=WzNIJlWsPKwW

Link Video Presentasi:

https://www.youtube.com/watch?v=ClqDZUCRxO0