Credit Card Default Prediction

Group 10
Zihao Li, Lunjing Yuan, Haorui Cheng, Mengyao Song

Background information

- In financial industry, credit score is a common risk control method. It uses personal information and data submitted by credit card applicants to predict the probability of future defaults and credit card borrowings. The bank is able to decide whether to issue a credit card to the applicant.

- Logistic model is a common method for credit scoring. Logistic is suitable for binary classification tasks and can calculate the coefficients of each feature.

- More predictive methods including KNeighbors, Random Forest, Decision Tree and Support Vector Machines have been introduced into credit card scoring.

Purpose of our analysis

Building machine learning models to predict if the applicant is a 'good' or 'bad' client.

End-to-end Machine Learning project

Here are the main steps we will go through:

- Get the data
- Data cleaning and preprocessing
- Discover and visualize the data to gain insights
- Select and train model
- Conclusions

Get the data

Data source: https://www.kaggle.com/rikdifos/credit-card-approval-prediction

Data description :

- Table 1: application_record

  Numerics: CNT_CHILDREN, AMT_INCOME_TOTA, DAYS_EMPLOYED , CNT_FAM_MEMBERS

  Categoricals:FLAG_OWN_CAR,FLAG_OWN_REALTY,NAME_INCOME_TYPE , NAME_EDUCATION_TYPE, NAME_FAMILY_STATUS ,NAME_HOUSING_TYPE, DAYS_BIRTH, FLAG_MOBIL, FLAG_WORK_PHONE, FLAG_PHONE, FLAG_EMAIL, OCCUPATION_TYPE

- Table 2: credit_record

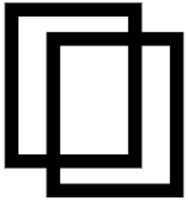
  Numerics: ID

  Categoricals: MONTHS_BALANCE, STATUS

Prepare the data for Machine Learning

Drop

Transfer



Merge

Label

Overall statistics of table 1 : application

	CNT_CHILDREN	AMT_INCOME_TOTAL	DAYS_BIRTH	DAYS_EMPLOYED	FLAG_MOBIL	FLAG_WORK_PHONE	FLAG_PHONE	FLAG_EMAIL	CNT_FAM_MEMBER
count	101658.000000	1.016580e+05	101658.000000	101658.000000	101658.0	101658.000000	101658.000000	101658.000000	101657.00000
mean	0.434073	1.858836e+05	-16066.993803	63155.300449	1.0	0.216451	0.300271	0.086545	2.20356
std	0.739729	1.046114e+05	4204.728138	141002.279846	0.0	0.411827	0.458378	0.281169	0.90986
min	0.000000	2.700000e+04	-25201.000000	-16365.000000	1.0	0.000000	0.000000	0.000000	1.00000
25%	0.000000	1.192500e+05	-19575.000000	-3109.000000	1.0	0.000000	0.000000	0.000000	2.00000
50%	0.000000	1.575000e+05	-15624.000000	-1493.000000	1.0	0.000000	0.000000	0.000000	2.00000
75%	1.000000	2.250000e+05	-12564.000000	-357.000000	1.0	0.000000	1.000000	0.000000	3.00000
max	19.000000	3.825000e+06	-7489.000000	365243.000000	1.0	1.000000	1.000000	1.000000	20.00000

Drop duplicated rows

```
application.loc[application.DAYS_EMPLOYED==-1194].loc[application.DAYS_BIRTH== -10554]
```

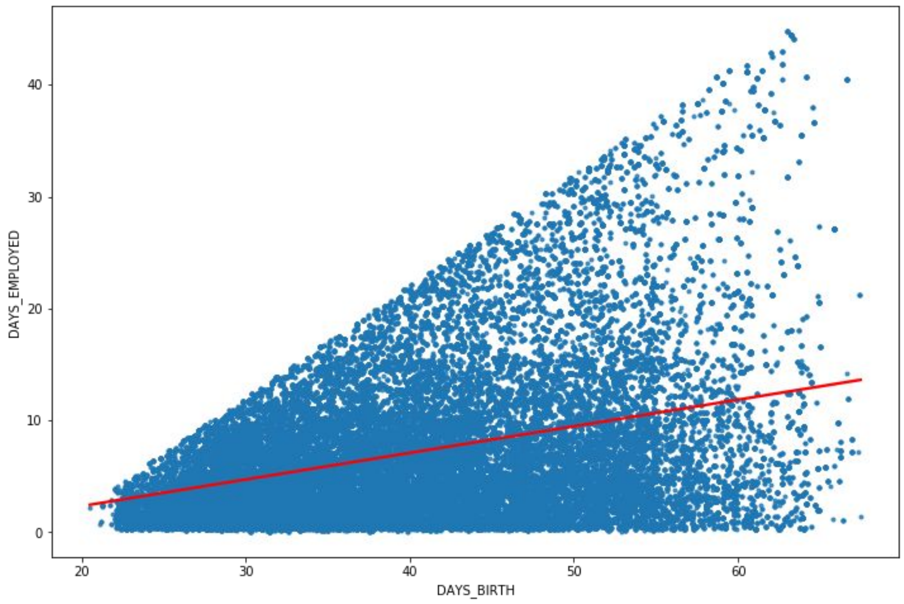
	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYPE
213	5009031	F	N	Y	0	315000.0	Working	Secondary / secondary special
214	5009032	F	N	Y	0	315000.0	Working	Secondary / secondary special
215	6153669	F	N	Y	0	315000.0	Working	Secondary / secondary special

```
application = application.drop_duplicates(subset=application.columns[1:], keep='first', inplace=False)
application.loc[application.DAYS_BIRTH== -10554]
```

	ID	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	NAME_INCOME_TYPE	NAME_EDUCATION_TYP
213	5009031	F	N	Y	0	315000.0	Working	Secondary / seconda speci
359960	6491536	F	Y	Y	0	234000.0	Working	Secondary / seconda speci
363841	6508956	M	Y	Y	0	157500.0	Working	Secondary / seconda speci
375944	6590652	M	Y	N	0	315000.0	Commercial associate	Secondary / seconda speci

Employed_days : outlier and resacling

Outlier:
emplyed_days=**365243**




```
app['employed_years'] = app[['age', 'employed_years']].apply(
    lambda x :lm.predict(np.array([x['age']]).reshape(1,-1))[0]
    if pd.isna(x['employed_years']) else x['employed_years'], axis=1)
```

CNT_CHILDREN /CNT_FAM_MEMBERS : outliers

```
application.CNT_CHILDREN.value_counts()
```

```
0     304071
1      88527
2      39884
3       5430
4        486
5        133
7          9
9          5
6          4
12         4
14         3
19         1
```

SET:
CNT_CHILDREN < 10

Mobile

```
application.FLAG_MOBIL.value_counts()
```

```
1    90085
Name: FLAG_MOBIL, dtype: int64
```

Overall statistics of table 2

```
credit.head()
```

	ID	MONTHS_BALANCE	STATUS
0	5001711	0	X
1	5001711	-1	0
2	5001711	-2	0
3	5001711	-3	0
4	5001712	0	C

```
df['decline'] = df.STATUS.apply(lambda x : 1 if x in (4, 5) else 0)
```

ID: Unique Id of the row in application record.

MONTHS_BALANCE: The number of months from record time.

STATUS: Credit status for this month.

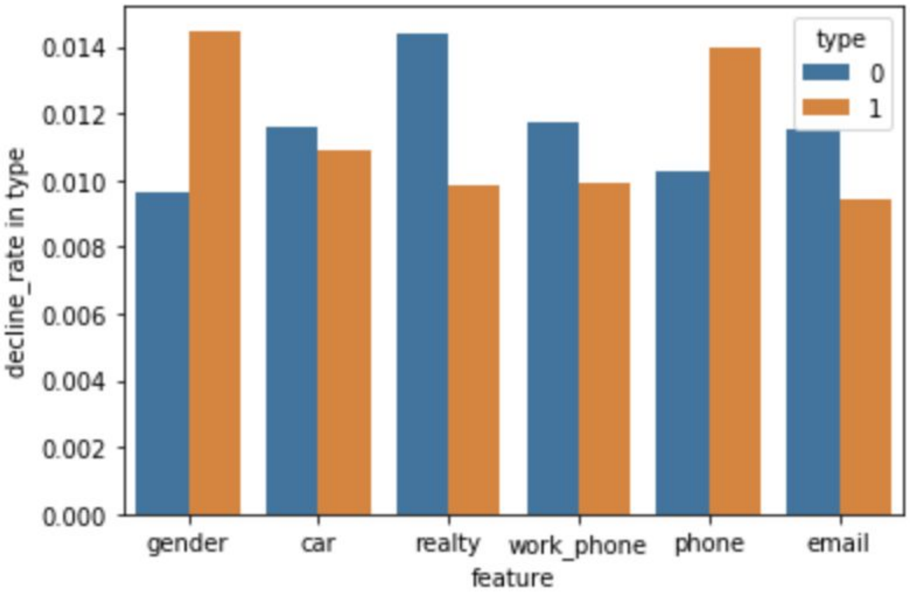
```
  X: No loan for the month
  C: paid off that month
  0: 1-29 days past due
  1: 30-59 days past due
  2: 60-89 days overdue
  3: 90-119 days overdue
  4: 120-149 days overdue
  5: Overdue or bad debts, write-offs for more
than 150 days
```

Create Labels

- 0: Decline =< 3 months or pay off on time
- 1: Decline > 3 months

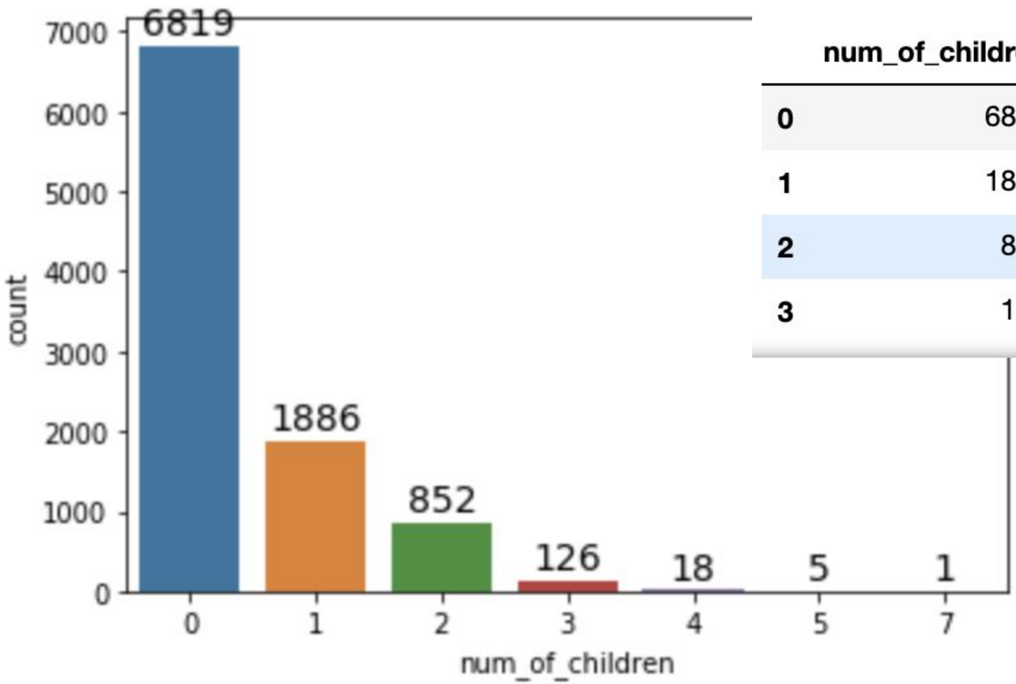
Data Visualization

Binary features



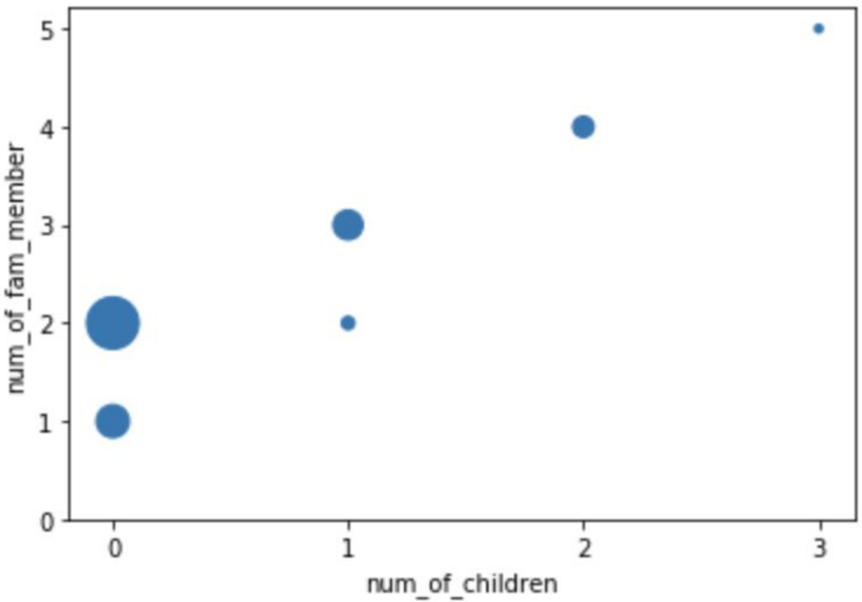
	feature	type	decline_rate in type	count	decline_count
0	gender	0	0.009649	6322	61
1	gender	1	0.014476	3385	49
2	car	0	0.011567	6138	71
3	car	1	0.010927	3569	39
4	realty	0	0.014425	3189	46
5	realty	1	0.009819	6518	64
6	work_phone	0	0.011715	7597	89
7	work_phone	1	0.009953	2110	21
8	phone	0	0.010268	6915	71
9	phone	1	0.013968	2792	39
10	email	0	0.011516	8857	102
11	email	1	0.009412	850	8

Digital features: number of child



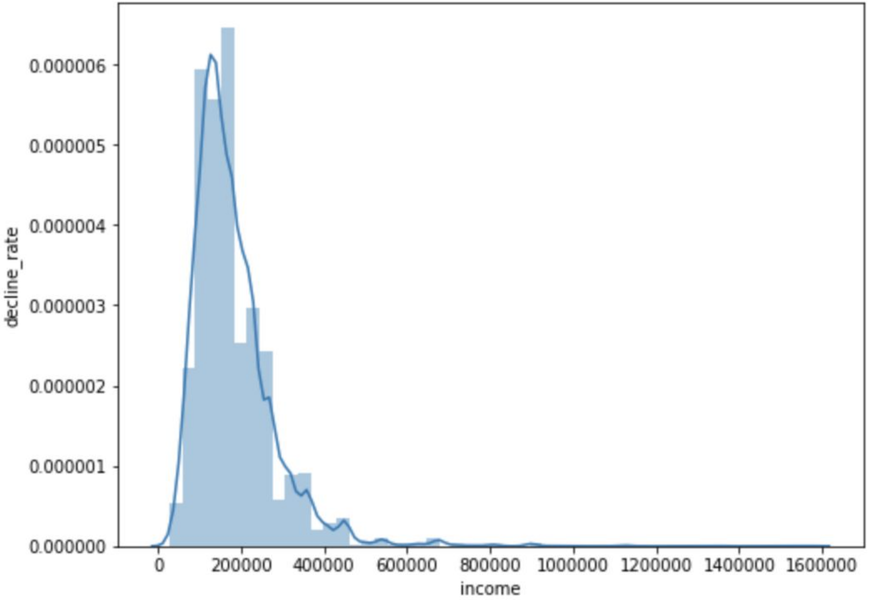
	num_of_children	decline_count	approved_count	decline_rate
0	6819	75	6744	0.010999
1	1886	25	1861	0.013256
2	852	9	843	0.010563
3	150	1	149	0.006667

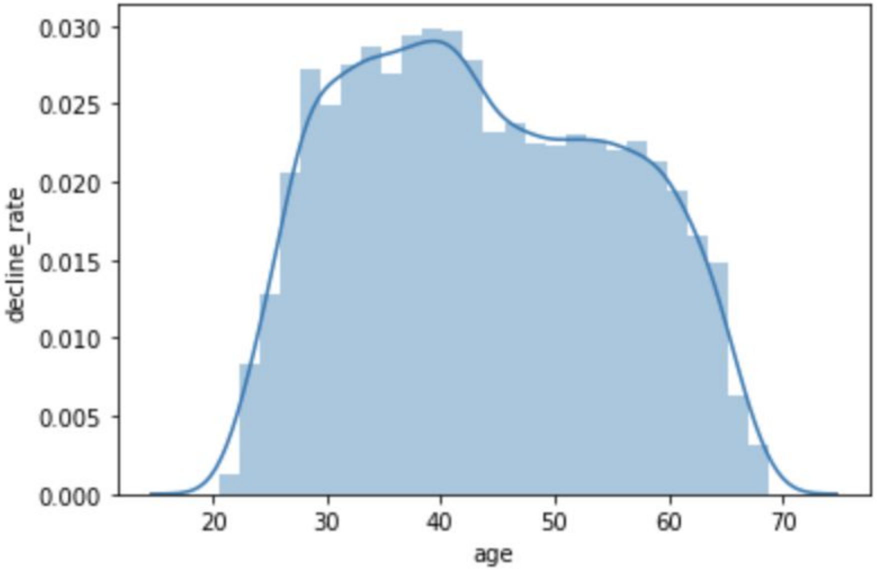
Digital features : number of children with number of family members



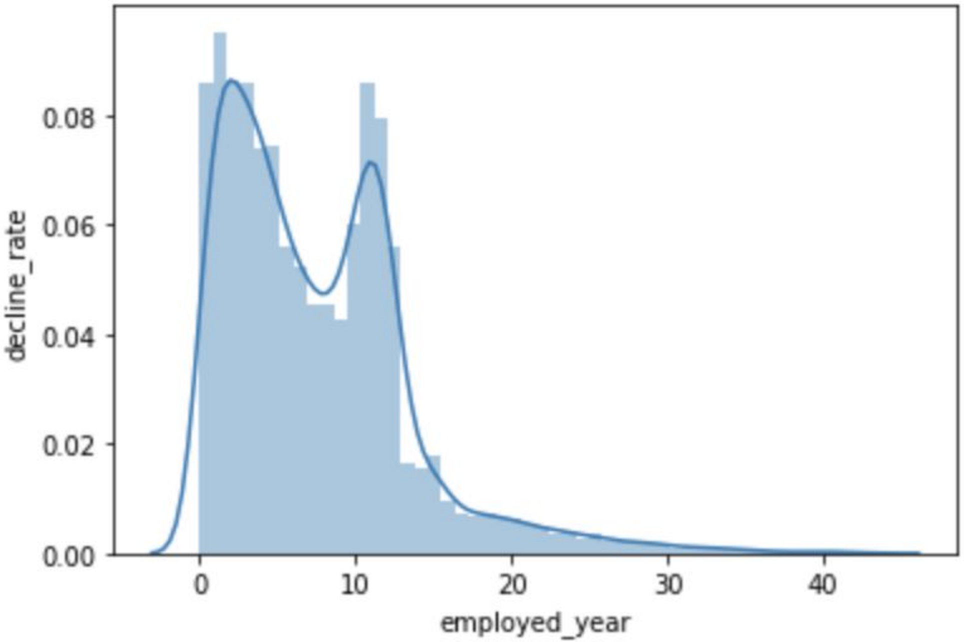
	num_of_children	num_of_fam_member	times
1	0	2	4880
0	0	1	1939
4	1	3	1577
7	2	4	792
3	1	2	303
9	3	5	116

Digital features : Total income and Age

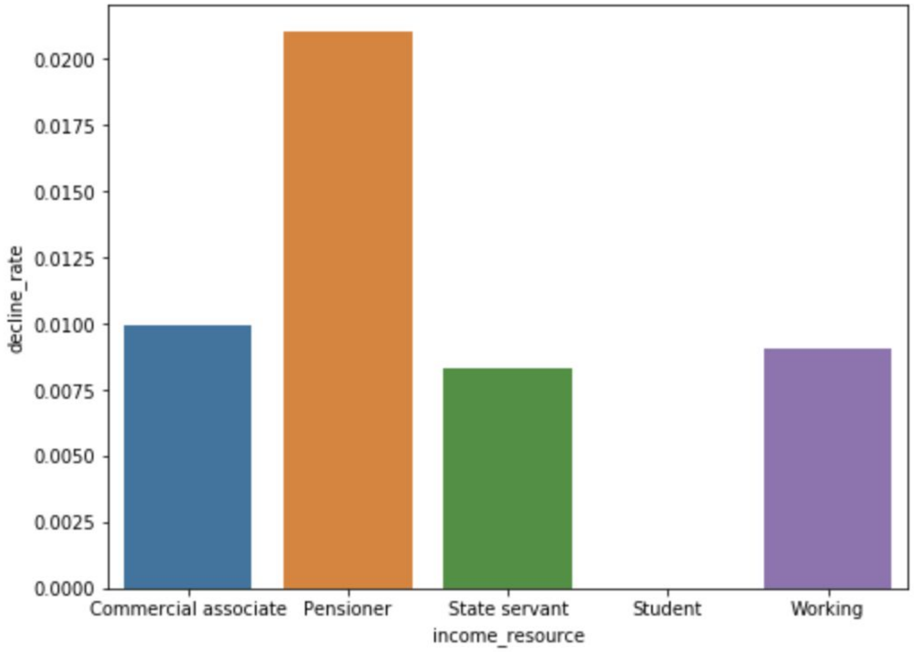


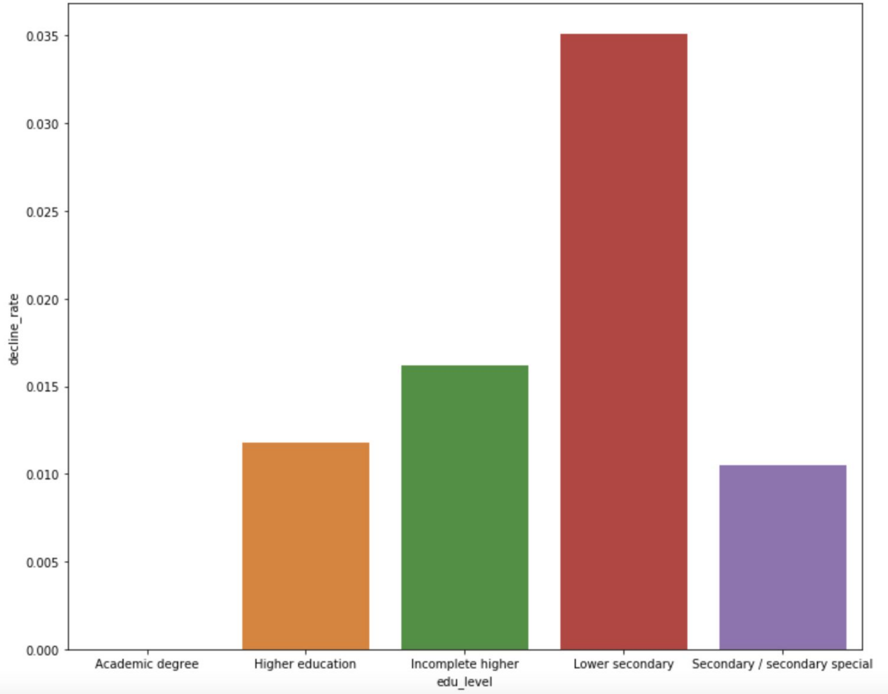


Employed years

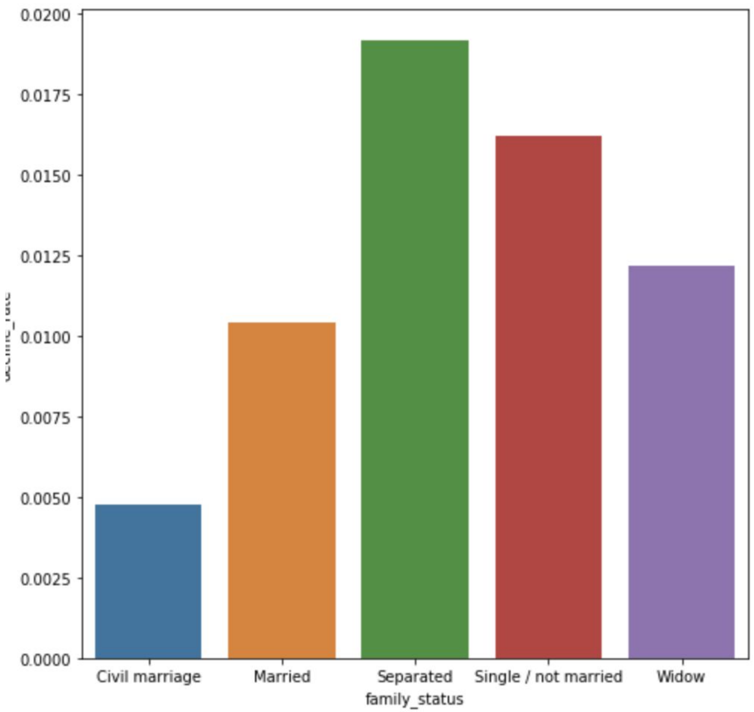


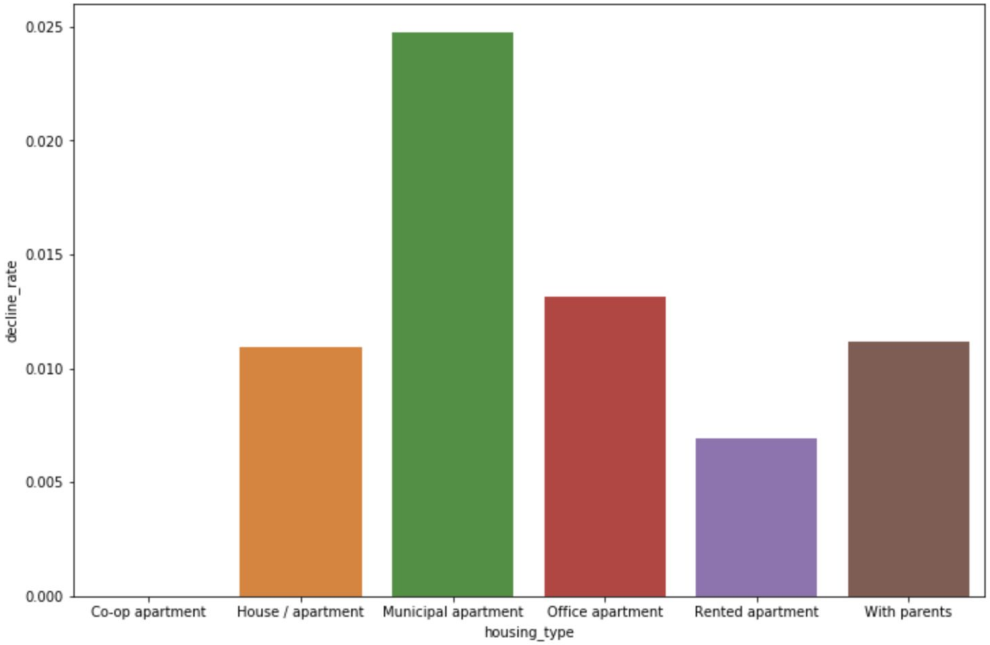
Class feature: income_resource/education level



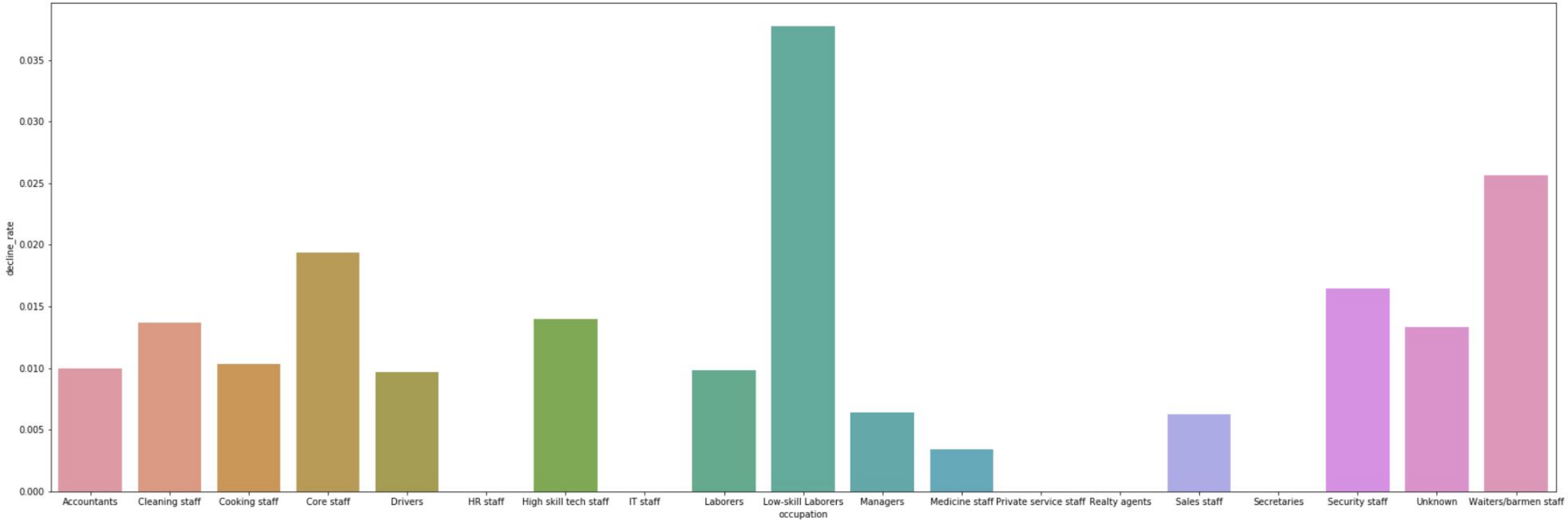


Family status / housing type





Occupation type



Learning by doing

Concept of IV & WOE

Information Value (IV):

- **A measure of how well a variable is able to distinguish between a binary response to the target variable.**

$$IV = \sum (\%\text{ of non-events} - \%\text{ of events}) * WOE$$

Information Value	Variable Predictiveness
Less than 0.02	Not useful for prediction
0.02 to 0.1	Weak predictive Power
0.1 to 0.3	Medium predictive Power
0.3 to 0.5	Strong predictive Power
>0.5	Suspicious Predictive Power

Concept of IV & WOE

Weight of Evidence(WoE):

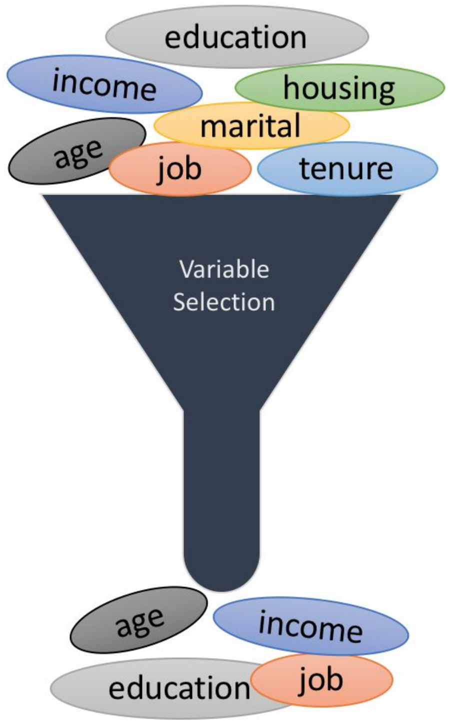
- **Predictive power of the independent variable in relation to the dependent variable**

$$WOE = \ln\left(\frac{\text{Distribution of Goods}}{\text{Distribution of Bads}}\right)$$

```
def calc_iv(df, target, feature):
    lst = []
    for i in range(df[feature].nunique()):
        val = list(df[feature].unique())[i]
        total = df[df[feature] == val].count()[feature]
        Good = df[(df[feature] == val) & (df[target] == 0)].coun
        Bad = df[(df[feature] == val) & (df[target] == 1)].count
        lst.append([feature, val, total, Good, Bad])

    temp = pd.DataFrame(lst, columns=['Var', 'Value', 'Total', '
    temp['Share'] = temp['Total'] / temp['Total'].sum()
    temp['Bad Rate'] = temp['Bad'] / temp['Total']
    temp['Good_distri'] = temp['Good'] / temp['Good'].sum()
    temp['Bad_distri'] = temp['Bad'] / temp['Bad'].sum()
    temp['WoE'] = np.log(temp['Good_distri'] / temp['Bad_distri'
    temp['IV'] = temp['WoE'] * (temp['Good_distri'] - temp['Bad_

    temp = temp.sort_values(by=['Var', 'Value'], ascending=[True
    temp.index = range(len(temp.index))

    iv = round(temp['IV'].sum(),8)
    if ((iv == np.inf) | (iv == -np.inf)):
```

```
calc_iv(df_copy, 'decline', 'phone')
```

```
0.02108813
```



Problem : Imbalance Data

```
y_train.value_counts()
```

```
0    6650
1     144
Name: decline, dtype: int64
```

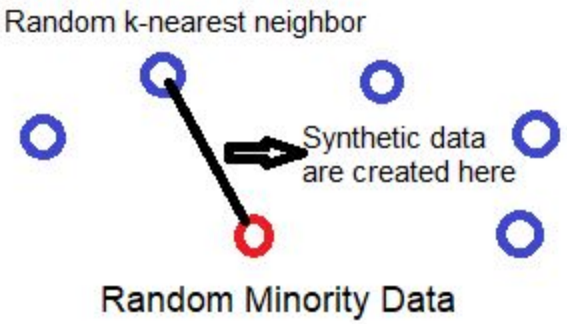
```
0    6650
1    6650
Name: decline, dtype: int64
```

SMOTE Techniques On Oversampling For Imbalance Data

- SMOTE only works for continuous features
- In our case, we use SMOTE-NC for mixed (categorical and continuous) features

Cited From: https://towardsdatascience.com/5-smote-techniques-for-oversampling-your-imbalance-data-b8155bdbe2b5

Introduction To SMOTE



Cited From:

https://towardsdatascience.com/5-smote-techniques-for-oversampling-your-imbalance-data-b8155bdbe2b5

Select models to train

Machine Learning Algorithms

- Logistic Regression
- Decision Tree
- Random Forest
- KNeighbors
- Ridge
- Adaboost
- Voting

Accuracy Score

```
--------------------------------------------------------------------------------
accuracy:
Best params:
{'classifier': RidgeClassifier(alpha=1, max_iter=3000), 'classifier__alpha': 1, 'classifier__max_iter': 3000, 'preprocessing': StandardScaler()}

Best cross-validation score: 0.94
Test-set score: 0.91
========================================================
accuracy:
Best params:
{'classifier': LogisticRegression(C=100, max_iter=1000), 'classifier__C': 100, 'classifier__max_iter': 1000, 'preprocessing': StandardScaler()}

Best cross-validation score: 0.96
Test-set score: 0.93
========================================================
accuracy:
Best params:
{'classifier': DecisionTreeClassifier(criterion='entropy', max_depth=14), 'classifier__criterion': 'entropy', 'classifier__max_depth': 14, 'preprocessing': N
one}

Best cross-validation score: 0.96
Test-set score: 0.92
========================================================
```

Recall

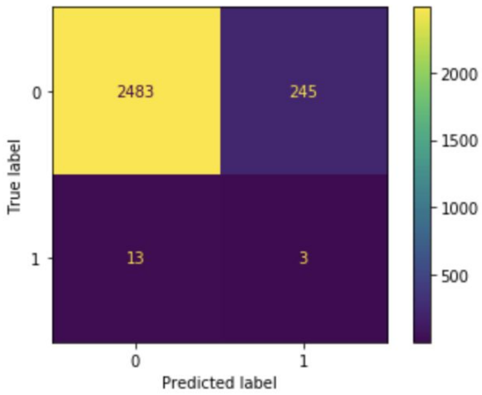
```
----------------------------------------------------------------------------------------------------
recall:
Best params:
{'classifier': RidgeClassifier(alpha=1, max_iter=3000), 'classifier__alpha': 1, 'classifier__max_iter': 3000, 'preprocessing': StandardScaler()}

Best cross-validation score: 0.97
Test-set score: 0.19
==================================================
recall:
Best params:
{'classifier': LogisticRegression(C=100, max_iter=1000), 'classifier__C': 100, 'classifier__max_iter': 1000, 'preprocessing': StandardScaler()}

Best cross-validation score: 0.98
Test-set score: 0.06
==================================================
recall:
Best params:
{'classifier': DecisionTreeClassifier(max_depth=2), 'classifier__criterion': 'gini', 'classifier__max_depth': 2, 'preprocessing': None}

Best cross-validation score: 1.00
Test-set score: 0.44
==================================================
```

ROC_AUC Score

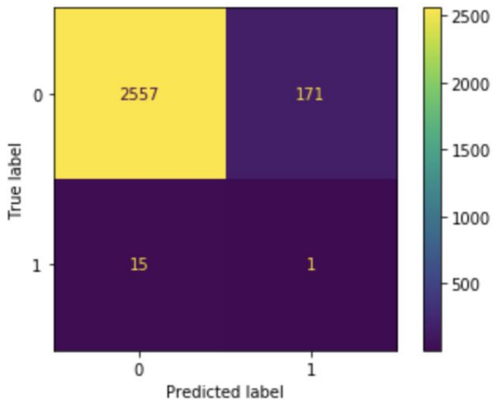
```
------------------------------------------------------------------------------------------------------------------------------------------
roc_auc:
Best params:
{'classifier': RidgeClassifier(alpha=91, max_iter=3000), 'classifier__alpha': 91, 'classifier__max_iter': 3000, 'preprocessing': StandardScaler()}

Best cross-validation score: 0.99
Test-set score: 0.48
=====================================================================
roc_auc:
Best params:
{'classifier': LogisticRegression(C=100, max_iter=1000), 'classifier__C': 100, 'classifier__max_iter': 1000, 'preprocessing': StandardScaler()}

Best cross-validation score: 0.99
Test-set score: 0.48
=====================================================================
roc_auc:
Best params:
{'classifier': DecisionTreeClassifier(criterion='entropy', max_depth=14), 'classifier__criterion': 'entropy', 'classifier__max_depth': 14, 'preprocessing': N
one}

Best cross-validation score: 0.98
Test-set score: 0.54
=====================================================================
------------------------------------------------------------------------------------------------------------------------------------------
```

Confusion Matrix

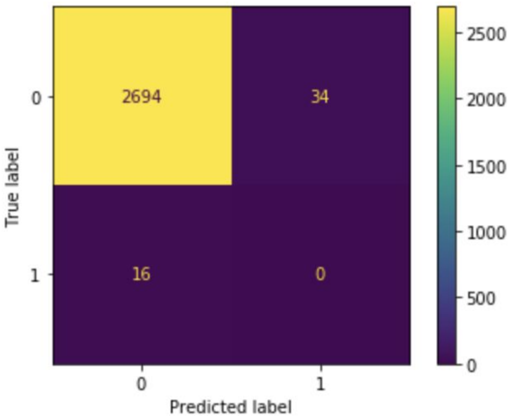
current clf: Ridge
test data score: 0.1875



current clf: LogisticRegression
test data score: 0.0625



current clf: DecisionTree
test data score: 0.0



Conclusion

1. **Accuracy Score:** Ridge Regression
2. **ROC_AUC Score & Recall Score** (future work)