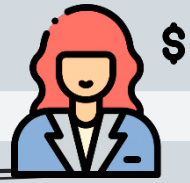
BDS

防貸少年團！

109034511 邱綉雅



目錄

背景介紹

-5W1H-

背景介紹

WHY	借貸公司資金有限，在貸款時難以評估是否應貸款給貸款人
HOW	建立模型來評估顧客是否能夠償還貸款
WHAT	借貸評估模型
WHO	借貸公司
WHERE	申辦貸款處
WHEN	借貸公司審查貸款人資料時

資料分析
&
前處理

資料分析


```
Data columns (total 27 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   loan_amnt             396030 non-null  float64
 1   term                  396030 non-null  object
 2   int_rate              396030 non-null  float64
 3   installment           396030 non-null  float64
 4   grade                 396030 non-null  object
 5   sub_grade             396030 non-null  object
 6   emp_title             373103 non-null  object
 7   emp_length            377729 non-null  object
 8   home_ownership        396030 non-null  object
 9   annual_inc            396030 non-null  float64
 10  verification_status   396030 non-null  object
 11  issue_d               396030 non-null  object
 12  loan_status           396030 non-null  object
 13  purpose               396030 non-null  object
 14  title                 394275 non-null  object
 15  dti                   396030 non-null  float64
 16  earliest_cr_line      396030 non-null  object
 17  open_acc              396030 non-null  float64
 18  pub_rec               396030 non-null  float64
 19  revol_bal             396030 non-null  float64
 20  revol_util            395754 non-null  float64
 21  total_acc             396030 non-null  float64
 22  initial_list_status   396030 non-null  object
 23  application_type      396030 non-null  object
 24  mort_acc              358235 non-null  float64
 25  pub_rec_bankruptcies  395495 non-null  float64
 26  address               396030 non-null  object
```

```
df.isnull().sum()
```

loan_amnt	0
term	0
int_rate	0
installment	0
grade	0
sub_grade	0
emp_title	22927
emp_length	18301
home_ownership	0
annual_inc	0
verification_status	0
issue_d	0
loan_status	0
purpose	0
title	1755
dti	0
earliest_cr_line	0
open_acc	0
pub_rec	0
revol_bal	0
revol_util	276
total_acc	0
initial_list_status	0
application_type	0
mort_acc	37795
pub_rec_bankruptcies	535
address	0
loan_repaid	0

資料前處理

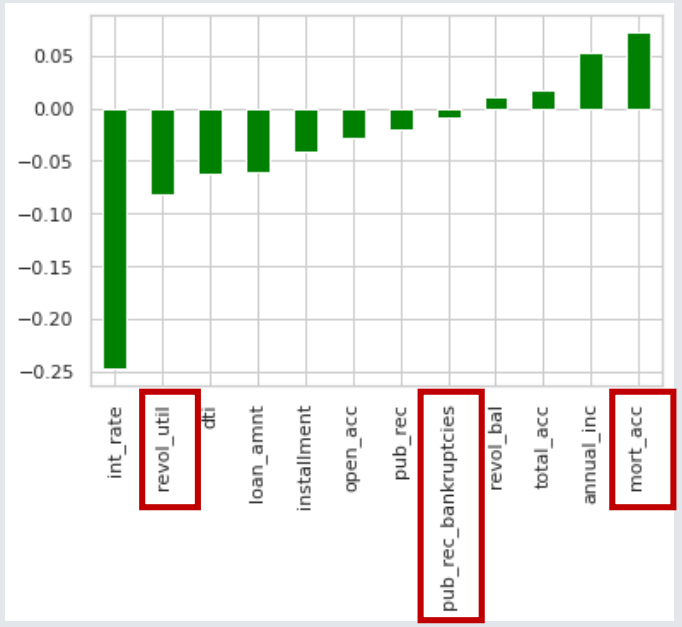
資料前處理

- 創建一個loan_repaid的欄位，取代原本屬於類別資料的loan_status

```
df['loan_repaid'] = df['loan_status'].map({'Fully Paid':1,'Charged Off':0})
df[['loan_repaid','loan_status']].head()
```

- 找出其他資料欄位和loan_repaid的相關性

```
df.corr()['loan_repaid'].sort_values(ascending=True).drop('loan_repaid').plot.bar(color='green')
```




```
df.isnull().sum()
```

loan_amnt	0
term	0
int_rate	0
installment	0
grade	0
sub_grade	0
emp_title	22927
emp_length	18301
home_ownership	0
annual_inc	0
verification_status	0
issue_d	0
loan_status	0
purpose	0
title	1755
dti	0
earliest_cr_line	0
open_acc	0
pub_rec	0
revol_bal	0
revol_util	276
total_acc	0
initial_list_status	0
application_type	0
mort_acc	37795
pub_rec_bankruptcies	535
address	0
loan_repaid	0

資料前處理

- Drop

```
print(df['emp_title'].nunique())
df['emp_title'].value_counts()
```

```
173105
Teacher                          4389
Manager                          4250
Registered Nurse                 1856
RN                               1846
Supervisor                       1830
                                 ...
CLK Properties                      1
driver / sales                      1
Aspire Technologies                 1
crabiel inc                         1
Financial advisor associate         1
Name: emp_title, Length: 173105, dtype: int64
```

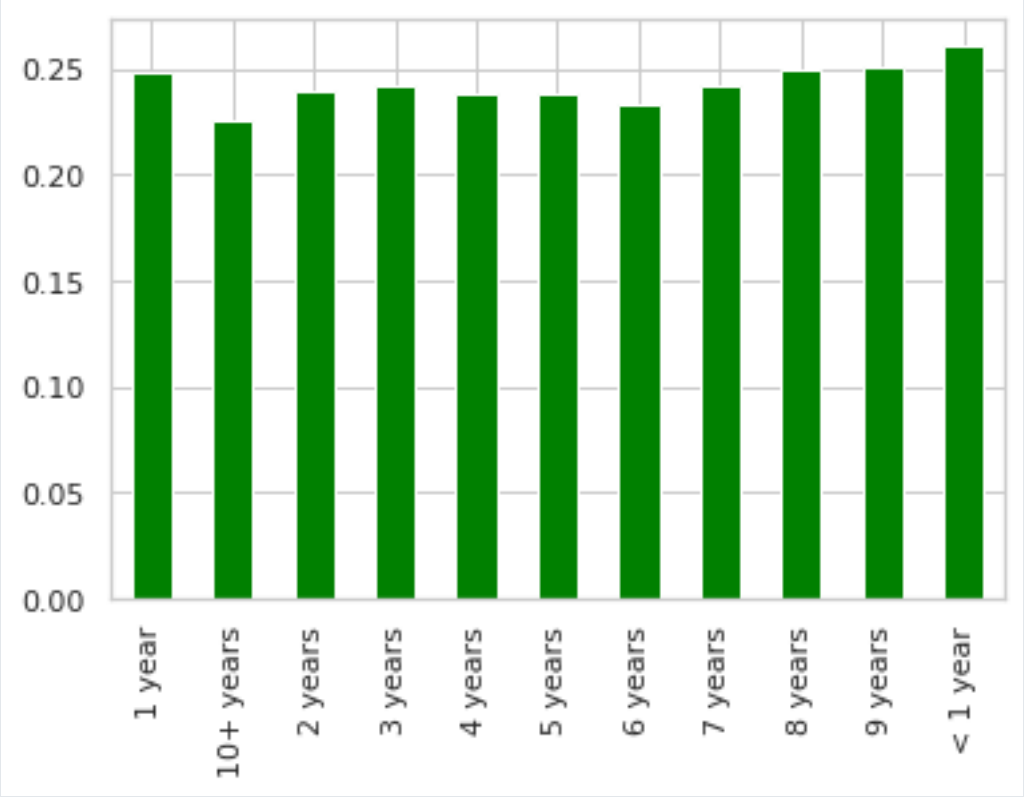
```
df[['title',  'purpose']].head(10)
```

	title	purpose
0	Vacation	vacation
1	Debt consolidation	debt_consolidation
2	Credit card refinancing	credit_card
3	Credit card refinancing	credit_card
4	Credit Card Refinance	credit_card
5	Debt consolidation	debt_consolidation
6	Home improvement	home_improvement
7	No More Credit Cards	credit_card
8	Debt consolidation	debt_consolidation
9	Debt Consolidation	debt_consolidation

資料前處理

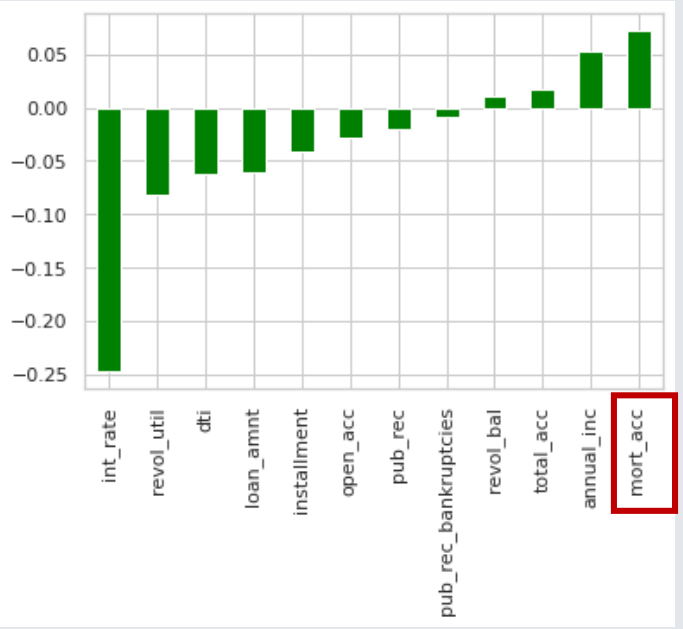
- Drop

```
per_charge_off = df[df['loan_repaid'] == 0]['emp_length'].value_counts() / df[df['loan_repaid'] == 1]['emp_length'].value_counts()
per_charge_off.plot.bar(color='green')
```



資料前處理

- 補值




```
print("Mean of mort_acc column per total_acc")
total_acc_avg = df.groupby('total_acc').mean()['mort_acc']
print(total_acc_avg)
```

```
total_acc_avg = df.groupby('total_acc').mean()['mort_acc']

def fill_mort_acc(total_acc,mort_acc):
    '''
    Accepts the total_acc and mort_acc values for the row.
    Checks if the mort_acc is NaN , if so, it returns the avg mort_acc value
    for the corresponding total_acc value for that row.

    total_acc_avg here should be a Series or dictionary containing the mapping of the
    groupby averages of mort_acc per total_acc values.
    '''
    if np.isnan(mort_acc):
        return total_acc_avg[total_acc]
    else:
        return mort_acc

df['mort_acc'] = df.apply(lambda x: fill_mort_acc(x['total_acc'], x['mort_acc']), axis=1)
```

資料前處理

- Drop

```
df = df.dropna()
```

```
df.isnull().sum()
```

loan_amnt	0
term	0
int_rate	0
installment	0
grade	0
sub_grade	0
emp_title	22927
emp_length	18301
home_ownership	0
annual_inc	0
verification_status	0
issue_d	0
loan_status	0
purpose	0
title	1755
dti	0
earliest_cr_line	0
open_acc	0
pub_rec	0
revol_bal	0
revol_util	276
total_acc	0
initial_list_status	0
application_type	0
mort_acc	37795
pub_rec_bankruptcies	535
address	0
loan_repaid	0

```
df.isnull().sum()
```

loan_amnt	0
term	0
int_rate	0
installment	0
grade	0
sub_grade	0
home_ownership	0
annual_inc	0
verification_status	0
issue_d	0
purpose	0
dti	0
earliest_cr_line	0
open_acc	0
pub_rec	0
revol_bal	0
revol_util	0
total_acc	0
initial_list_status	0
application_type	0
mort_acc	0
pub_rec_bankruptcies	0
address	0
loan_repaid	0

資料前處理

- 類別轉數值

① apply

```
print(df['term'].value_counts())
print('\n')
print('\n')

df['term'] = df['term'].apply(lambda term: int(term[:3]))

print(df['term'].value_counts())
```

```
 36 months    301247
 60 months     93972
Name: term, dtype: int64



36    301247
60     93972
Name: term, dtype: int64
```

② get_dummies

```
df['home_ownership'] = df['home_ownership'].replace(['NONE', 'ANY'], 'OTHER')
dummies = pd.get_dummies(df['home_ownership'],drop_first=True)
df = df.drop('home_ownership',axis=1)
df = pd.concat([df,dummies],axis=1)
```

資料前處理

- 切割資料集

```
#split  the  dataset
#  Features
X  =  df.drop('loan_repaid',axis=1).values

#  Label
y  =  df['loan_repaid'].values

X  =  np.asarray(X).astype('float32')
y  =  np.asarray(y).astype('float32')
#  Split
X_train,  X_test,  y_train,  y_test  =  train_test_split(X,  y,  test_size=0.2,  random_state=101)
X_train,  X_val,  y_train,  y_val  =  train_test_split(X_train,  y_train,  test_size=0.2,  random_state=1)
```

資料前處理

- 資料正規化

```
scaler = MinMaxScaler()

# fit and transfrom
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_val = scaler.transform(X_val)

# everything has been scaled between 1 and 0
print('Max: ',X_train.max())
print('Min: ', X_train.min())
```

模型建立
&
方法介紹

模型建立

- NN

```
#build  a  model
model  =  Sequential()

#  input  layer
model.add(Dense(78,activation='relu'))
model.add(Dropout(0.2))

#  hidden  layer
model.add(Dense(39,activation='relu'))
model.add(Dropout(0.2))

#  hidden  layer
model.add(Dense(19,activation='relu'))
model.add(Dropout(0.2))

#  output  layer
model.add(Dense(1,  activation='sigmoid'))

#  compile  model
model.compile(optimizer="adam",  loss='binary_crossentropy',metrics=['accuracy'])
```

參數調整

optimizer	結果	選擇
adam	0.888	ⓥ
sgd	0.887	
RMSprop	0.887	
adagrad	0.887	

loss function	結果	選擇
mean_squared_error	0.888	
binary_crossentropy	0.888	ⓥ
squared_hinge	0.845	
logcosh	0.888	

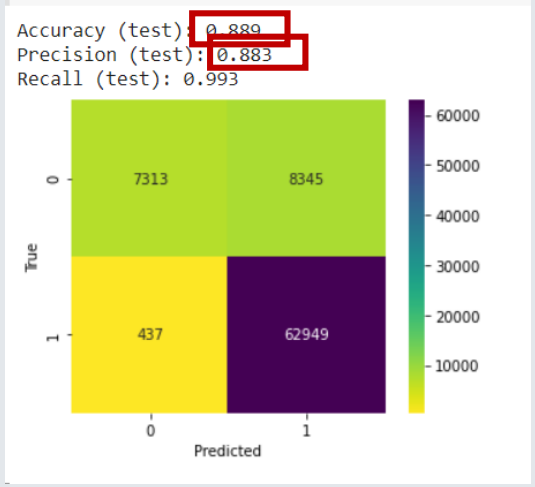
Input+hidden layer activation function	結果	選擇
relu	0.888	ⓥ
tanh	0.888	
exponential	0.198	
linear	0.887	

output layer activation function	結果	選擇
sigmoid	0.888	ⓥ
tanh	0.887	
softmax	0.802	
relu	0.888	

模型建立

- Catbooster

```
from catboost import CatBoostClassifier, Pool
pool_train = Pool(X_train, y_train)
pool_test = Pool(X_test, y_test)
pool_val = Pool(X_val, y_val)
model = CatBoostClassifier(learning_rate=0.03,
                           iterations=1000,
                           early_stopping_rounds=100,
                           verbose=False,
                           random_state=0)
model.fit(pool_train, eval_set=pool_val, plot=True);
```



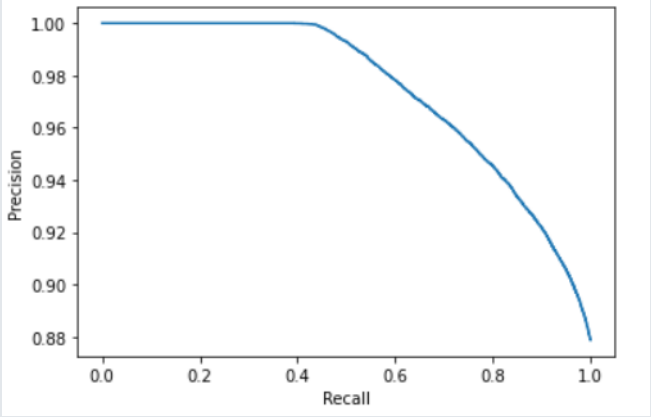

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix, precision_recall_curve
y_pred_test = model.predict(pool_test)

acc_test = accuracy_score(y_test, y_pred_test)
prec_test = precision_score(y_test, y_pred_test)
rec_test = recall_score(y_test, y_pred_test)
print(f'''Accuracy (test): {acc_test:.3f}
Precision (test): {prec_test:.3f}
Recall (test): {rec_test:.3f}''')

cm = confusion_matrix(y_test, y_pred_test)
ax = sns.heatmap(cm, cmap='viridis_r', annot=True, fmt='d', square=True)
ax.set_xlabel('Predicted')
ax.set_ylabel('True');
```

模型改善

```
import pandas.util.testing as tm
```



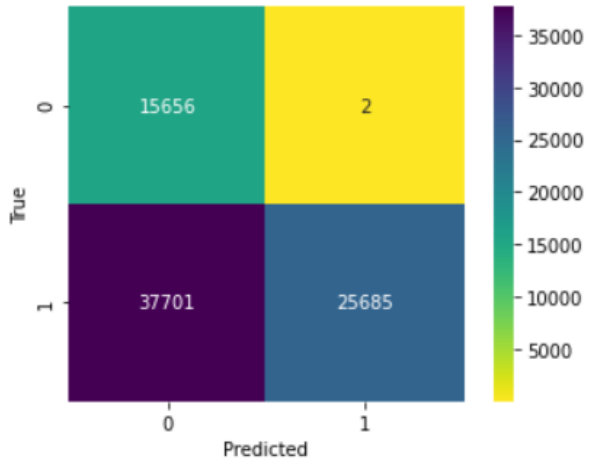

```
y_proba_val = model.predict_proba(pool_val)[:, 1]
p_val, r_val, t_val = precision_recall_curve(y_val, y_proba_val)
plt.plot(r_val, p_val)
plt.xlabel('Recall')
plt.ylabel('Precision');
```

```
p_max = p_val[p_val != 1].max()
t_all = np.insert(t_val, 0, 0)
t_adj_val = t_all[p_val == p_max]
y_adj_val = (y_proba_val > t_adj_val).astype(int)
p_adj_val = precision_score(y_val, y_adj_val)
print(f'Adjusted precision (validation): {p_adj_val:.3f}')
```

```
y_proba_test = model.predict_proba(pool_test)[:, 1]
y_adj_test = (y_proba_test > t_adj_val).astype(int)
p_adj_test = precision_score(y_test, y_adj_test)
r_adj_test = recall_score(y_test, y_adj_test)
print(f'''Adjusted precision (test): {p_adj_test:.3f}
Adjusted recall (test): {r_adj_test:.3f}''')

cm_test = confusion_matrix(y_test, y_adj_test)
ax = sns.heatmap(cm_test, cmap='viridis_r', annot=True, fmt='d', square=True)
ax.set_xlabel('Predicted')
ax.set_ylabel('True');
```

```
Adjusted precision (test): 1.000
Adjusted recall (test): 0.405
```



結論
&
未來展望

結論

- 結論

  兩模型比較後，可以看出Catboost在改善前和NN模型的準確率差異不大，但經過模型改善後，因為Catboost可選擇僅最小化偽陽性誤差，因此能夠有較好的預測準確率。

- 未來展望

  可以考慮納入其餘考量因素，如：疫情期間，一些風險高的工作從業者可能較難還出貸款，或是針對企業貸款和個人貸款做分類研究，使預測結果更加精確。

Q & A