DATA@ANZ PROGRAM

PREDICTIVE ANALYTICS IN PYTHON BY GUSTAVO MONTENEGRO

The methodology

First, I filtered the dataset by the 'pay/salary' transactions. Then, the salary was calculated with the sum of the total amount per customer. This calculation considers all the transactions since august to october.

Not everyone has salary transactions by month. This is the reason why I decided to make the sum with the three months period.

The annual salary is four times the sum of the total amount.

```
Salary_df= df[df['txn_description'] == 'PAY/SALARY']

Salary_df[['customer_id','amount']].groupby('customer_id').sum()
```

I used the 'Salary_df' dataframe in order to make calculations grouped by 'customer_id'. I made a correlations with the features that I already had using the 'corr ( )' function with the default method: pearson.

The results

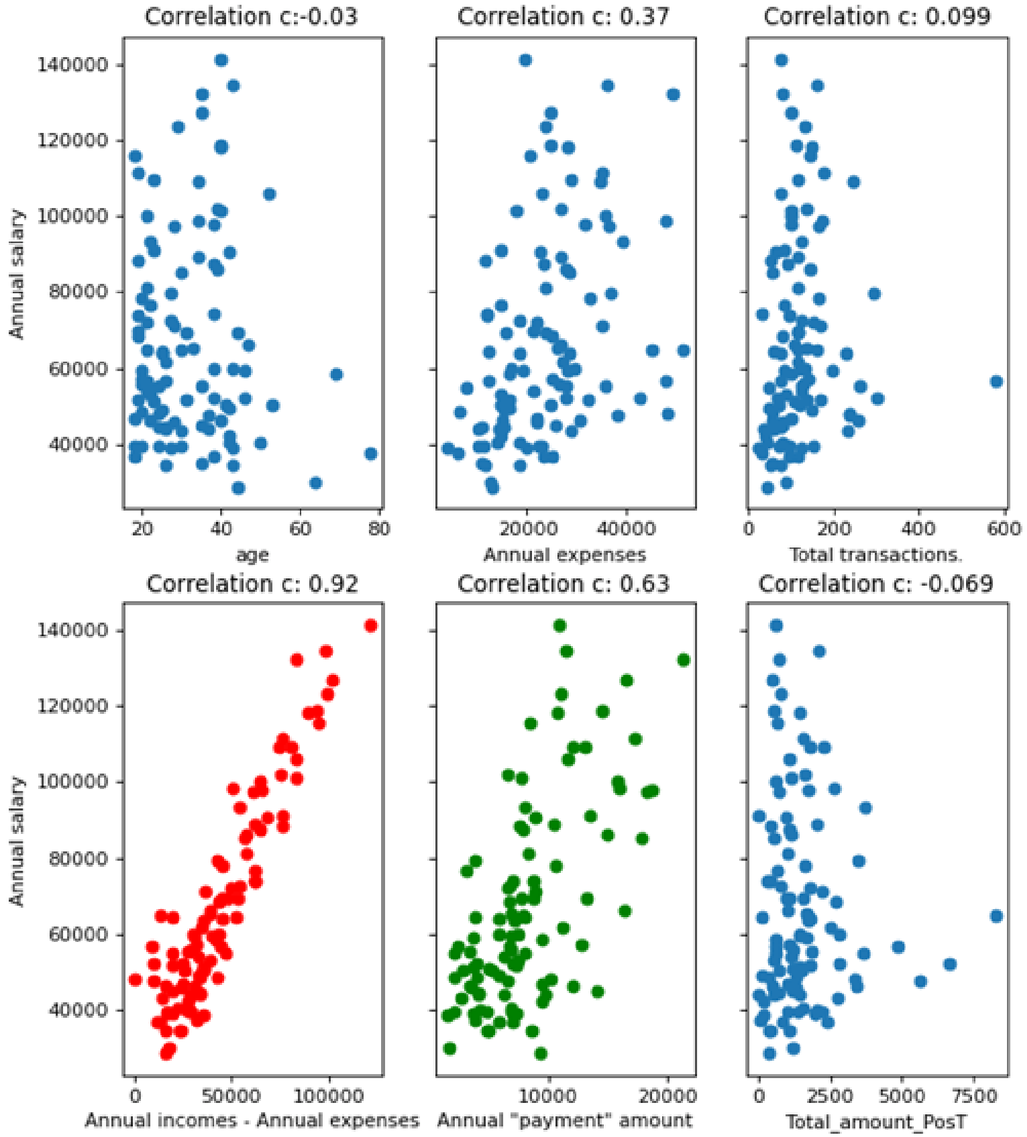
With some financial concepts I found the next useful features in order to predict the annual salary.

Annual incomes - Annual expenses = **Discretionary income**

The annual expenses were calculated with the sum of total debit movements per customer. Then, four times this total, like the annual salary. The annual incomes are equal to the annual salary.

Total amount of **payment transactions** by year

Usually when the incomes increase, some kind of expenses increase too. That was how I found this feature.



The linear regression model

The model performance is good with these two features. I used the 'score' measure in order to evaluate the accuracy.

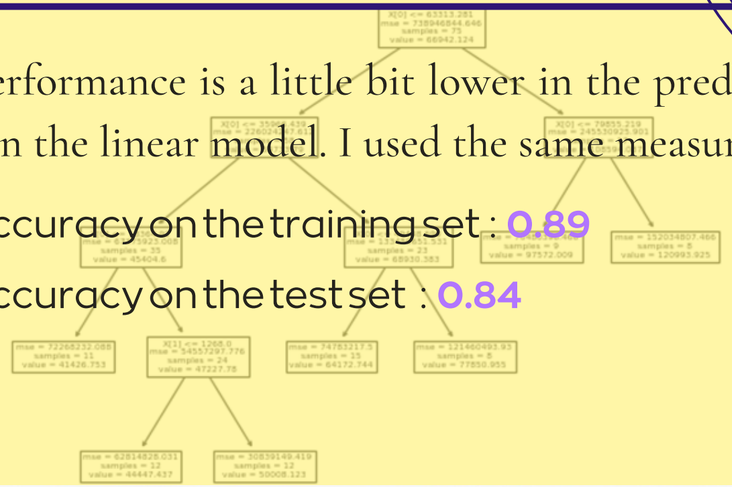
The accuracy on the training set : **0.88**

The accuracy on the test set : **0.89**

I think this model could be used like a fast approach.

```
LR_df = Qsalary1[['A incomes - expenses','Total_amount_PT','AS']]
X = LR_df[['A incomes - expenses','Total_amount_PT']]
y = LR_df['AS']
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state = 0)
Linear_model = LinearRegression().fit(X_train,y_train)
#print('Linear model coeff (w): {}'
     #.format(Linear_model.coef_))
#print('Linear model intercept (b): {:.3f}'
     #.format(Linear_model.intercept_))
print('Annual salary regression model')
print('Accuracy of linear regression model on training set: {:.3f}'
     .format(Linear_model.score(X_train, y_train)))
print('Accuracy of linear regression model on test set: {:.3f}'
     .format(Linear_model.score(X_test, y_test)))
```

The decision tree regressor



The performance is a little bit lower in the prediction set tnan the linear model. I used the same measure.

The accuracy on the training set : **0.89**

The accuracy on the test set : **0.84**

```
DT_df = Qsalary1[['A incomes - expenses','Total_amount_PT','AS']]
#binned = pd.cut(DT_df['AS'], bins = [0,23000,90800,405000,1000000],
                #labels = ['L','L-M','M-H','H'] )
#DT_df['AS_b'] = binned

X = DT_df[['A incomes - expenses','Total_amount_PT']]
y = DT_df['AS']

from sklearn.tree import DecisionTreeRegressor
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    random_state = 0)

clf = DecisionTreeRegressor(max_depth = 4,
                            min_samples_leaf = 8,random_state
                            = 0).fit(X_train, y_train)
print('Annual salary decision tree')
print('Accuracy of DT classifier on training set: {:.2f}'
     .format(clf.score(X_train, y_train)))
print('Accuracy of DT classifier on test set: {:.2f}'
     .format(clf.score(X_test, y_test)))
```