Homework 7

your name and id

In this homework, imagine that you've already become a data scientist hired by the bank. Your job is to help the bank to decide whether approve the loan applications from customers. Certain customers may have higher risk to result in the loan default, and your job is to distinguish them and suggest your manager not to approve their application from the very beginning.

The data you have is the record of many previous loan cases (including the information of applicants) and their final outcome (default or non-defualt in the loan). Specifically, there's one column called `loan_status` , **where the value 0 is non-default and 1 denotes default.** You may find more information about this dataset here (https://www.kaggle.com/laotse/credit-risk-dataset).

Task 1: Load the data and Pre-processing

1. Use `pandas` to read the csv file named `credit_risk_dataset.csv` as dataframe (The file is in the Canvas page of this homework).

1. Drop three columns, named `person_home_ownership` , `loan_intent` , `loan_grade` .

1. Drop all the samples with missing values. hint: you can use the `dropna` method, see documentation here (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dropna.html).

1. For the column named `cb_person_default_on_file` , replace "Y" with integer 1, and "N" with 0. Hint: there are many ways to do this, and you can refer to one possible tutorial here (https://www.geeksforgeeks.org/replace-the-column-contains-the-values-yes-and-no-with-true-and-false-in-python-pandas/)

1. Name your final pre-processed dataframe as `df_credit`

```
In [ ]: # write your code here
```

Check your answers with the following codes. Your results should be the same as printed below. (You may notice that there are some unrealisitc values, but don't worry about them in the basic tasks. Resolve this issue in optional task if you like to).

```
In [4]: df_credit
```

Out[4]:

	person_age	person_income	person_emp_length	loan_amnt	loan_int_rate	loan_status	loan_percent_income
0	22	59000	123.0	35000	16.02	1	0.59
1	21	9600	5.0	1000	11.14	0	0.10
2	25	9600	1.0	5500	12.87	1	0.57
3	23	65500	4.0	35000	15.23	1	0.53
4	24	54400	8.0	35000	14.27	1	0.55
...	...	...	...	...	...	...	...
32576	57	53000	1.0	5800	13.16	0	0.11
32577	54	120000	4.0	17625	7.49	0	0.15
32578	65	76000	3.0	35000	10.99	1	0.46
32579	56	150000	5.0	15000	11.48	0	0.10
32580	66	42000	2.0	6475	9.99	0	0.15

28638 rows × 9 columns

```
In [6]: df_credit.describe()
```

Out[6]:

	person_age	person_income	person_emp_length	loan_amnt	loan_int_rate	loan_status	loan_percent_i
count	28638.000000	2.863800e+04	28638.000000	28638.000000	28638.000000	28638.000000	28638.0
mean	27.727216	6.664937e+04	4.788672	9656.493121	11.039867	0.216600	0.1
std	6.310441	6.235645e+04	4.154627	6329.683361	3.229372	0.411935	0.1
min	20.000000	4.000000e+03	0.000000	500.000000	5.420000	0.000000	0.0
25%	23.000000	3.948000e+04	2.000000	5000.000000	7.900000	0.000000	0.0
50%	26.000000	5.595600e+04	4.000000	8000.000000	10.990000	0.000000	0.1
75%	30.000000	8.000000e+04	7.000000	12500.000000	13.480000	0.000000	0.2
max	144.000000	6.000000e+06	123.000000	35000.000000	23.220000	1.000000	0.8

```
In [9]: df_credit.isnull().sum() # check the nan numbers
```

```
Out[9]: person_age                    0
        person_income                 0
        person_emp_length             0
        loan_amnt                     0
        loan_int_rate                 0
        loan_status                   0
        loan_percent_income           0
        cb_person_default_on_file     0
        cb_person_cred_hist_length    0
        dtype: int64
```

Task 2 : Loan Status Prediction with Logistic Regression

1. Pick up column named `loan_status` , convert it to the numpy array `y` . Convert the dataset containing the remaining columns as numpy array `X` .

1. Create `X_train` , `X_test` , `y_train` , `y_test` by splitting the dataset. The test dataset should consist of 33% percent of the whole data. Hint: you can use the train_test_split (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html) function in sklearn.

1. Train a logistic regression model **with sklearn** (you can directly call the functions/methods in sklearn) on training dataset. Report the performance on test dataset

In [ ]:
```
# write your code here
```

Task 3: Prediction for a New Customer

1. Your bank now receive a new loan application. Load the information about this new customer in the csv file `new_customer_info.csv` (The file is in the Canvas page of this homework).

1. Based on your model, what is the predicted loan status (default or non-default) of this new customer? **Optional**: before making the prediction, you may choose to update your model by re-train the parameter with all the data in `X` and `y` (this step is called "finalize" in PyCaret), to avoid wasting the test dataset in Task 2.

1. Because of the pandemic, your bank becomes more conservative about approving new loans. The manager requires that the loan can only be approved if the risk (probability) of default is below 15%. Based on your model, what is your suggested decision for this loan application? Write the code and one short paragraph (in Markdown) telling your manager 1)the decision you suggest 2) the reasons why and 3) description of all the hard works you have done to make this suggestion, i.e. how you built this model. Note that your manager has not learned machine learning previously, so please try to explain in plain language.

Hint: you may find the `predict_proba` method in the logistic regression classifier helpful to solve the third problem.

In [39]:

```
from sklearn.linear_model import LogisticRegression
help(LogisticRegression.predict_proba)
```

```
Help on function predict_proba in module sklearn.linear_model._logistic:

predict_proba(self, X)
    Probability estimates.

    The returned estimates for all classes are ordered by the
    label of classes.

    For a multi_class problem, if multi_class is set to be "multinomial"
    the softmax function is used to find the predicted probability of
    each class.
    Else use a one-vs-rest approach, i.e calculate the probability
    of each class assuming it to be positive using the logistic function.
    and normalize these values across all the classes.

    Parameters
    ----------
    X : array-like of shape (n_samples, n_features)
        Vector to be scored, where `n_samples` is the number of samples and
        `n_features` is the number of features.

    Returns
    -------
    T : array-like of shape (n_samples, n_classes)
        Returns the probability of the sample for each class in the model,
        where classes are ordered as they are in ``self.classes_``.
```

In [ ]:

```
# write your code here
```

write your message to manager here

Task 4: Your final project

This is not a task in homework, but a reminder that you can already finish Task 1 and 2 in final project. Start it right NOW and don't wait until the last minute.

Optional Task

1. It's a pitty that we drop many categorical columns in the original data. You can follow this notebook (https://www.kaggle.com/zhaoyunma/credit-risk-prediction) to pre-process all the variables. Will this help improve your model?
2. Try the classification module (https://github.com/pycaret/pycaret/blob/master/tutorials/Binary%20Classification%20Tutorial%20Level%20Beginner%20-%20%20CLF101.ipynb) in PyCaret for this dataset.

In [ ]:

```
# write your code here
```