Credit Scoring: Exam 01

Complete Name:
Github Username:
Student ID:
Date:

Contents

Part 1: (40 pts) Theory

(6 pts) - Credit Risk Theory

The following equation describes the expected loss for a credit operation:

$$EL = PD * EAD * LGD$$

Where:

- EL: expected loss
- PD: probability of default
- EAD: exposure at default
- LGD: loss given default

(1.5 pts) When doing a credit risk model, which variable are we interested in forecasting?

(1.5 pts) What are the "units" for the EL and EAD?

(3 pts) In your own words, explain the concept of LGD.
Hint: remember that LGD is a percentage (%).

(7 pts) - Credit Scoring Performance

We can use two strategies to improve credit-scoring performance:

- **Strategy 1**: Increase data quality, availability, and volume (e.g., big data)
- **Strategy 2**: Use more complex modeling techniques (e.g., non-linear models)

(3.5 pts) In your own words, explain the disadvantages of only using **strategy 1**.

(3.5 pts) In your own words, explain the disadvantages of only using **strategy 2**.

(7 pts) - Data Quality Dimensions

(2 pts) In your own words, explain the importance of the `SEC` (Security) dimension on financial data:

(2 pts) Give an example of data that's sensible to the `TIM` (Timeliness) dimension. Elaborate:

(3 pts) What's the difference between the `COM` (Completeness) and `AC` (Accuracy) data dimensions? Give an example.

(10 pts) - Object Oriented Programming

(0.5 pts for each) Name the 4 object-oriented principles:

-
-
-
-

(2.5 pts for each) Explain 2 of the object-oriented principles:

-
-

(1.5 pts) Describe the difference between objects and classes:

(1.5 pts) Describe the difference between attributes and methods:

(10 pts) - Classes/Objects in Python

Consider the following `Human` class with two sub-classes named `Student` and `Professor`.

`Human` class definition:

```
import datetime as dt

STRING_FORMAT_DATE = "%Y-%m-%d"


class Human:

    def __init__(self, first_name, last_name, date_of_birth, **kwargs):
        self.date_of_birth = dt.datetime.strptime(date_of_birth, STRING_FORMAT_DATE)
        self.first_name = first_name
        self.last_name = last_name
        self.full_name = f"{first_name} {last_name}"
        self._kwargs = kwargs

    @property
    def age(self):
        today, dob = dt.datetime.today(), self.date_of_birth
        adjust = (today.month, today.day) < (self.dob.month, self.dob.day)
        return today.year - self.dob.year - adjust


    def greeting(self):
        raise NotImplementedError("Greeting method is not implemented")
```

Child class `Student` definition:

```python
class Student(Human):
    notebook_name = "notes"

    @property
    def notes(self):
        return self._kwargs.get(self.notebook_name, "")

    def add_note(self, note):
        notes_content = self.notes + note
        self._kwargs[self.notebook_name] = notes_content

    def greeting(self):
        return "My name is {student_name} and I'm {student_age} years old.".format(
            student_name=self.full_name, student_age=self.age)
```

Child class `Professor` definition:

```python
class Professor(Human):

    @property
    def lecture(self):
        return self._kwargs.get("lecture")

    def assign_lecture(self, lecture_name, override=False, fail=True):
        FAIL_MESSAGE = f"Cannot assign lecture {lecture_name} to professor " + \
                       f"{self.full_name} because {self.lecture} was previously assigned."
        if not self.lecture or override:
            self._kwargs["lecture"] = lecture_name
        elif not fail:
            print(FAIL_MESSAGE)
        else:
            raise ValueError(FAIL_MESSAGE)

    def greeting(self):
        return "I'm Prof. {professor_last_name} and {lecture_details}.".format(
            professor_last_name=self.last_name,
            lecture_details=f"I am teaching a lecture named '{self.lecture}'"
                if self.lecture else "I am currently not teaching any lecture")
```

Name the attributes, and methods of the following classes:

- (2 pts) `Student`
  - Attributes:
  - Methods:
- (2 pts) `Professor`
  - Attributes:
  - Methods:

Given this code snippet:

```
# Create professor object
professor = Professor(
    first_name="Erwin",
    last_name="Schrödinger",
    date_of_birth="1887-08-12"
)

# A) First greeting
greeting_a = professor.greeting()

# B) Second greeting
professor.assign_lecture(lecture_name="Quantum Mechanics", fail=False)
greeting_b = professor.greeting()

# C) Third greeting
professor.assign_lecture(lecture_name="Probability Theory", fail=False)
greeting_c = professor.greeting()
```

Write out the value of the following variables:

- (2 pts) Value of `greeting_a`:
- (2 pts) Value if `greeting_c`:

(2 pts) What's a python-decorator and why is it used? List and explain each element.

Part 2: (60 pts) Practice

1. Go to the following github repository:
    - https://github.com/rhdzmota/958d7822-da73-4bbc-817f-fa79ac0778bc
2. Fork the repository in github.
3. Clone the fork into your local machine using git-bash.
4. Change directory into the repository.
    - `cd 958d7822-da73-4bbc-817f-fa79ac0778bc`
5. There are 2 standalone python applications:
    - `cashflows`
    - `monty-hall`
6. Solve the problems and push your changes to the master branch.
    - `git status`
    - `git add path/to/file.py`
    - `git commit -m "commit message"`
    - `git push origin master`

(30 pts) - Cashflows

1. Use pycharm or your favorite code-editor to open the `cashflows` python project.
2. Read the project documentation on the `README.md` file.
3. Create a python virtual-environment (`venv`)
4. Activate the virtual-environment.
    - Linux/Mac: `source venv/bin/activate`
    - Windows: `source venv/Scripts/activate`
5. Install dependencies with `pip install -r requirements.txt`

Cashflow

Create a `Cashflow` class on `util.py` with the following:

- (1.5 pts) Attributes:
    - `amount` - monetary amount at time t.
    - `t` - integer representing time.
- (1.5 pts) Methods:
    - `present_value` - given an interest-rate as an argument, calculate the present value of the cashflow.

Investment project

(6 pts) Implement the `plot` method on the `InvestmentProject` class and `plot_investment` on the `Main` class.

- Only show the plot (i.e., `plt.show()`) when the `show` argument is true.
- Save the plot as a png-file when `save` contains a filename.
- The plot should have `x=t` and `y=amount`. Add title and axis labels.

(6 pts) Implement the `net_present_value` method on the `InvestmentProject` class.

- Remember that the NPV is the sum of all cashflows at time 0.
- If `interest_rate` is None, use object's `hurdle_rate`.

(6 pts) Implement the `equivalent_annuity` method on the `InvestmentProject` class.

- Remember that $P = A\frac{1-(1+i)^{-n}}{i}$.
- If `interest_rate` is None, use object's `hurdle_rate`.
- Use the net present value as `P`.

(1.5 pts) Successful execution of the following command:

```
$ python main.py plot_investment --filepath data/cashflows-1.csv --save file-1.png
```

(1.5 pts) Successful execution of the following command:

```
$ python main.py plot_investment --filepath data/cashflows-2.csv --show
```

(1.5 pts) Successful execution of the following command:

```
$ python main.py describe_investment --filepath data/cashflows-1.csv
  {
      "irr": 0.2205891139852516,
      "hurdle-rate": 0.08,
      "net-present-value": 653.7191648116468,
      "equivalent-annuity": 163.72818430111846
  }
```

(1.5 pts) Successful execution of the following command:

```
$ python main.py describe_investment --filepath data/cashflows-2.csv --hurdle-rate 0.02
{
    "irr": 0.059817182267134505,
    "hurdle-rate": 0.02,
    "net-present-value": 12.727618883755255,
    "equivalent-annuity": 3.3425750338217486
}
```

(1.5 pts) What does it means when the internal-rate of return is greater than the hurdle rate?

(1.5 pts) Can the net present value be negative? Why?

(30 pts) - Monty Hall

1. Use pycharm or your favorite code-editor to open the `monty-hall` python project.
2. Read the project documentation on the `README.md` file.
3. Create a python virtual-environment (`venv`)
4. Activate the virtual-environment.
   - Linux/Mac: `source venv/bin/activate`
   - Windows: `source venv/Scripts/activate`
5. Install dependencies with `pip install -r requirements.txt`

Single Game

(6 pts) Implement the "change" strategy on the `Guest` class defined in the private method `_choose_strategy_change`.

- Guest's final_choice should always be different than the first_choice and the reveal door when using the `change` strategy.
- The "single game" can receive: $opts > 2$.

(3 pts) Successful execution of the following command:

```
$ python main.py play_single_game \
    --strategy change \
    --opts 3
```

(3 pts) Successful execution of the following command:

```
$ python main.py play_single_game \
    --strategy change \
    --opts 7
```

Multiple games

(9 pts) Implement the `play_multiple_games` method on the `Main` class.

- Freq. probabilities must make sense.
- Use the `play_random_game` function from util.
- Should print the statistics on the console consistently.
- Should save the results only when passed the `--save` argument with a filename.

(3 pts) Successful execution of the following command:

```
$ python main.py play_multiple_games \
    --strategy random \
    --times 10000 \
    --opts 10 \
    --save file-1.csv
```

(3 pts) Successful execution of the following command:

```
$ python main.py play_multiple_games \
    --strategy change \
    --times 10000 \
    --opts 3 \
    --save file-2.csv
```

(3 pts) Successful execution of the following command:

```
$ python main.py play_multiple_games \
    --strategy stay \
    --times 10000 \
    --opts 3
```

(5 pts) - *Extra Points!*

(5 pts) Implement and apply decorator on the monty-hall problem:

- There's a decorator in the `util.py` file named: `remember_function_output_decorator`.
- Assume this decorator will only be used in `Guest` methods.
- Implementation: the decorator should automatically save the output of a method into the Guest's memory (list).
- Apply the decorator on the following methods. Modify as needed.
  - `_choose_strategy_random`
  - `_choose_strategy_change`
  - `_choose_strategy_stay`