Standardized Data Preparation User Study

In this study, you are a data scientist who is writing a data preparation (DP) script in two scenarios. You will be given a set of DP scripts to rate, according to how much you find them helpful to your DP work. You will rate the scripts on a scale between 1 ("Least representative/helpful") and 5 ("Most representative/helpful").

This study is conducted by Eugenie Lai, Yuze Lou, Brit Youngmann and Michael Cafarella. For any questions, feel free to contact Eugenie Lai by e-mail: eylai@mit.edu.

* Indicates required question

1. What is your full name? *

2. I have reviewed the consent form and agree to participate in this study. *

   Check all that apply.

   - [ ] I agree.

3. Are you a student in a Master's program?

   Mark only one oval.

   - ◯ Yes
   - ◯ No

4. Which of the following best describes your English ability?

   Mark only one oval.

   - ◯ Elementary Proficiency
   - ◯ Limited Working Proficiency
   - ◯ Minimum Professional Proficiency
   - ◯ Full Professional Proficiency
   - ◯ Native or Bilingual Proficiency

5. Number of years you have coded in Python

Mark only one oval.

- ◯ Less than 2
- ◯ Between 2 and 4
- ◯ More than 4

6. On a scale of 1-5, how familiar are you with the Pandas library?

Mark only one oval.

	1	2	3	4	5	
Hav	◯	◯	◯	◯	◯	Expert

Get familiar with the dataset, schema, and the modeling task.

You are a data scientist and decided to use the Titanic dataset to train a model to predict the survival label of the passengers in the test data. **The train/test data are .csv files that contain the following attributes (See the data dictionary below).**
Your model will be based on "features" like passengers' gender and class. You can also use feature engineering to create new features. In this user study, we only focus on the data preparation (DP) steps, which are Python code that makes any alternation to the data (in this case train.csv). In other words, we DO NOT consider code related to modeling (e.g., model selection, model evaluation).
Note: The scenario of this user study is adapted from the Kaggle Titanic competition. The study participant should be able to find all information needed in this Google form.

Data dictionary

Variable	Definition	Notes
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	A proxy for socio-economic status. 1st = Upper 2nd = Middle 3rd = Lower
sex	Sex	
Age	Age in years	Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5
sibsp	# of siblings / spouses aboard the Titanic	Sibling = brother, sister, stepbrother, stepsister Spouse = husband, wife (mistresses and fiancés were ignored)
parch	# of parents / children aboard the Titanic	Parent = mother, father Child = daughter, son, stepdaughter, stepson Some children travelled only with a nanny, therefore parch=0 for them.
ticket	Ticket number	
fare	Passenger fare	
cabin	Cabin number	
embarked	Port of embarkation	

Scenario 1: A cold start problem

It's your first time working with the Titanic dataset. You don't know where to start. Luckily, others have used the same dataset for their modeling task, and you have access to their DP scripts. You started to explore how other people process the dataset but quickly got overwhelmed by the volume of code and the necessary background knowledge you need to understand the code.

Now you think: It would be awesome if I could gather all common DP steps and combine them into a executable script as a starting point – let's call it a "magical starting script".

Given the top 10 most common DP steps over 60 scripts submitted by people participated in the Kaggle Titanic competition, rate the scripts based on the following:

1. On a scale between 1 ("Least representative") and 5 ("Most representative"), based on how much they are representative of the magical starting script.
2. On a scale between 1 ("Least helpful") and 5 ("Most helpful"), based on how helpful they are as a starting point for you as a data scientist working with the dataset for the first time.

Note: For each candidate script, if it includes any of the top 10 most common DP steps appeared in historical scripts, the corresponding line is highlighted in green.

Top 10 most common DP steps over 60 historical scripts

Rank	DP step	What it does
1	`X_train = df.drop('Survived', axis=1)`	Remove the target attribute from input training features
2	`y_train = df['Survived']`	Set the target attribute as label
3	`df['Age'] = df['Age'].fillna(df['Age'].mean())`	Fill NA/NaN values using mean age
4	`df['Embarked'] = df['Embarked'].fillna('S')`	Fill NA/NaN values using 'S'
5	`df['Embarked'] = df['Embarked'].map({'S': 0, 'C': 1, 'Q': 2}).astype(int)`	Map individual strings to a specific integer value
6	`df['Sex'] = df['Sex'].replace(['male', 'female'], [0, 1])`	Replace strings with integer values
7	`df['Fare'] = df['Fare'].fillna(df['Fare'].mean())`	Fill NA/NaN values using mean fare
8	`df['FamilySize'] = df['SibSp'] + df['Parch'] + 1`	Create a new feature using two existing features
9	`df = df.drop('Cabin', axis=1)`	Remove feature 'Carbin'
10	`df = df.drop(['Name'], axis=1)`	Remove feature 'Name'

7. Scenario 1: Script 1 representativeness

```python
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

# Load data
train = pd.read_csv('./data/input/titanic/train.csv')
# Data preprocessing
train['Age'].fillna(train['Age'].median(), inplace=True)
train['Embarked'].fillna(train['Embarked'].mode()[0], inplace=True)
train['FamilySize'] = train['SibSp'] + train['Parch'] + 1
train['IsAlone'] = 1
train['IsAlone'].loc[train['FamilySize'] > 0] = 0
encoder = OneHotEncoder()
train_encoded = pd.get_dummies(train, columns=['Sex', 'Embarked'])
```

Mark only one oval.

	1	2	3	4	5	
Leas	◯	◯	◯	◯	◯	Most representative

8. Scenario 1: Script 1 helpfulness

Mark only one oval.

	1	2	3	4	5	
Leas	◯	◯	◯	◯	◯	Most helpful

9. Scenario 1: Script 2 representativeness

```
import pandas as pd
import numpy as np

# Read the data
train = pd.read_csv('./data/input/titanic/train.csv')
# Data preprocessing
train['Age'] = train['Age'].fillna(train['Age'].mean())
train['Embarked'] = train['Embarked'].fillna('S')
train['family'] = train['SibSp'] + train['Parch'] + 1
train['Age'] = np.log(train['Age'] + 1)
train['Fare'] = np.log(train['Fare'] + 1)
```

Mark only one oval.

	1	2	3	4	5	
Leas	◯	◯	◯	◯	◯	Most representative

10. Scenario 1: Script 2 helpfulness

Mark only one oval.

	1	2	3	4	5	
Leas	◯	◯	◯	◯	◯	Most helpful

11. Scenario 1: Script 3 representativeness

```
import pandas as pd

# Read the data
train = pd.read_csv('./data/input/titanic/train.csv')
# Data preprocessing
train['Age'] = train['Age'].fillna(train['Age'].mean())
y = train['Survived']
X_train = train.drop('Survived', axis=1)
train['Sex'] = train['Sex'].replace(['male', 'female'], [0, 1], inplace=False)
train['Embarked'] = train['Embarked'].fillna('S', inplace=False)
train['Embarked'] = train['Embarked'].map({'S': 0, 'C': 1, 'Q': 2}).astype(int)
train['Fare'] = train['Fare'].fillna(train['Fare'].mean()), inplace=False)
```

Mark only one oval.

	1	2	3	4	5	
Leas	◯	◯	◯	◯	◯	Most representative

12. Scenario 1: Script 3 helpfulness

Mark only one oval.

	1	2	3	4	5	
Leas	◯	◯	◯	◯	◯	Most helpful

13. Scenario 1: Script 4 representativeness

```
import pandas as pd

# Read the data
train = pd.read_csv('./data/input/titanic/train.csv')
# Data preprocessing
train = pd.melt(train, id_vars=['Age', 'SibSp', 'Parch'], var_name='_unpivotVar',
value_name='_value')
train = train.pivot_table(index=['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Ticket',
'Fare', 'Cabin', 'Embarked'], columns="_unpivotVar", values="_value")
train = pd.melt(train, id_vars=['Age', 'Parch', 'SibSp'], var_name='_unpivotVar',
value_name='_value')
```

Mark only one oval.

	1	2	3	4	5	
Leas	◯	◯	◯	◯	◯	Most representative

14. Scenario 1: Script 4 helpfulness

Mark only one oval.

	1	2	3	4	5	
Leas	◯	◯	◯	◯	◯	Most helpful

15. Scenario 1: Without being given the most useful script above, how much time do you think you need to confidently compose an equivalent script **independently on your own**?

Mark only one oval.

- ◯ Less than 1 hour
- ◯ 1-2 hours
- ◯ 2-3 hours
- ◯ More than 3 hours

16. Scenario 1: Without being given the script you chose, but if you have access to the collection of 60 historical scripts, how much time do you think you need to confidently compose an equivalent script, **informed by the historical scripts**?

    Mark only one oval.

    - ◯ Less than 1 hour
    - ◯ 1-2 hours
    - ◯ 2-3 hours
    - ◯ More than 3 hours

Scenario 2: With a sketch DP script in mind

Now, in a more complex setting, imagine you are particularly interested in the survival rate for passengers in Class 3. You come up with an initial DP script, but you wonder if the DP steps you apply are the common ones used by other users. **Ideally, you hope to take advantage of the historical scripts, while achieving your own modeling goal** (i.e., focusing on Class 3 passengers).

Given the same top 10 most common DP steps over 60 scripts submitted by people participated in the Kaggle Titanic competition, rate the scripts based on the following:

1. On a scale between 1 ("Least representative") and 5 ("Most representative"), based on how much they are **representative of the common DP steps**.
2. On a scale between 1 ("Least helpful") and 5 ("Most helpful"), based on how helpful they are for you as a data scientist that hopes to **take advantage of the historical scripts, while focusing on Class 3 passengers**.

Note: For each candidate script, if it includes any of the top 10 most common DP steps appeared in historical scripts, the corresponding line is highlighted in green.

Top 10 most common DP steps over 60 historical scripts (same as Scenario 1)

Rank	DP step	What it does
1	`X_train = df.drop('Survived', axis=1)`	Remove the target attribute from input training features
2	`y_train = df['Survived']`	Set the target attribute as label
3	`df['Age'] = df['Age'].fillna(df['Age'].mean())`	Fill NA/NaN values using mean age
4	`df['Embarked'] = df['Embarked'].fillna('S')`	Fill NA/NaN values using 'S'
5	`df['Embarked'] = df['Embarked'].map({'S': 0, 'C': 1, 'Q': 2}).astype(int)`	Map individual strings to a specific integer value
6	`df['Sex'] = df['Sex'].replace(['male', 'female'], [0, 1])`	Replace strings with integer values
7	`df['Fare'] = df['Fare'].fillna(df['Fare'].mean())`	Fill NA/NaN values using mean fare
8	`df['FamilySize'] = df['SibSp'] + df['Parch'] + 1`	Create a new feature using two existing features
9	`df = df.drop('Cabin', axis=1)`	Remove feature 'Carbin'
10	`df = df.drop(['Name'], axis=1)`	Remove feature 'Name'

Your starting script

Note: In this scenario, you focus on evaluating how much the following scripts can **help you focus on Class 3 passengers, while incorporating the common DP steps**.

```
import pandas as pd
# Read the data
train = pd.read_csv('./data/input/titanic/train.csv')
# Data preprocessing
train = train[train['Pclass'] == 3]
train = train.drop(['Name', 'Ticket', 'Cabin', 'PassengerId'], axis=1)
train.Sex = train.Sex.map({'female': 0, 'male': 1})
median_age_men = train[train['Sex'] == 1]['Age'].median()
median_age_women = train[train['Sex'] == 0]['Age'].median()
train.loc[train.Age.isnull() & (train['Sex'] == 1), 'Age'] = median_age_men
train.loc[train.Age.isnull() & (train['Sex'] == 0), 'Age'] = median_age_women
train.Embarked = train.Embarked.map({'S': 0, 'C': 1, 'Q': 2, 'nan': 'NaN'})
train = train.dropna(inplace=False)
```

17. Scenario 2: Script 1 representativeness

Note: This script incorporates some common steps from the historical scripts, while preserving your modeling goal. Specifically, the script suggests that you handle the missing values for 'Age' and 'Embarked' differently, which are steps you didn't realize before.

```
import pandas as pd
# Read the data
train = pd.read_csv('./data/input/titanic/train.csv')
# Data preprocessing
train = train[train['Pclass'] == 3]
train = train.drop(['Name', 'Ticket', 'Cabin', 'PassengerId'], axis=1)
train.Sex = train.Sex.map({'female': 0, 'male': 1})
train['Age'] = train['Age'].fillna(train['Age'].mean())
train['Embarked'] = train['Embarked'].fillna('S', inplace=False)
train.Embarked = train.Embarked.map({'S': 0, 'C': 1, 'Q': 2, 'nan': 'NaN'})
train = train.dropna(inplace=False)
X = train.drop('Survived', axis=1)
y = train['Survived']
```

Mark only one oval.

	1	2	3	4	5	
Leas	◯	◯	◯	◯	◯	Most representative

18. Scenario 2: Script 1 helpfulness

Mark only one oval.

	1	2	3	4	5	
Leas	◯	◯	◯	◯	◯	Most helpful

19. Scenario 2: Script 2 representativeness

Note: This script appends two common steps from the historical scripts to your original script.

```
import pandas as pd
# Read the data
train = pd.read_csv('./data/input/titanic/train.csv')
# Data preprocessing
train = train[train['Pclass'] == 3]
train = train.drop(['Name', 'Ticket', 'Cabin', 'PassengerId'], axis=1)
train.Sex = train.Sex.map({'female': 0, 'male': 1})
median_age_men = train[train['Sex'] == 1]['Age'].median()
median_age_women = train[train['Sex'] == 0]['Age'].median()
train.loc[train.Age.isnull() & (train['Sex'] == 1), 'Age'] = median_age_men
train.loc[train.Age.isnull() & (train['Sex'] == 0), 'Age'] = median_age_women
train.Embarked = train.Embarked.map({'S': 0, 'C': 1, 'Q': 2, 'nan': 'NaN'})
train = train.dropna(inplace=False)
X = train.drop('Survived', axis=1)
y = train['Survived']
```

Mark only one oval.

	1	2	3	4	5	
Leas	◯	◯	◯	◯	◯	Most representative

20. Scenario 2: Script 2 helpfulness

Mark only one oval.

	1	2	3	4	5	
Leas	◯	◯	◯	◯	◯	Most helpful

21. Scenario 2: Script 3 representativeness

Note: This script is different from your starting script, which doe not preserve your modeling goal.

```
import pandas as pd
# Read the data
train = pd.read_csv('./data/input/titanic/train.csv')
# Data preprocessing
train['FamilySize'] = train['SibSp'] + train['Parch'] + 1
train['Title'] = train.Name.str.extract(' ([A-Za-z]+)\.', expand=False)
X = train.drop(['Survived', 'Name', 'Ticket', 'Cabin', 'PassengerId'], axis=1)
y = train['Survived']
```

Mark only one oval.

	1	2	3	4	5	
Leas	◯	◯	◯	◯	◯	Most representative

22. Scenario 2: Script 3 helpfulness

Mark only one oval.

	1	2	3	4	5	
Leas	◯	◯	◯	◯	◯	Most helpful

23. Scenario 2: Script 4 representativeness

Note: This script is identical to your starting script, which doesn't give you new knowledge about how to process the dataset.

```
import pandas as pd
# Read the data
train = pd.read_csv('./data/input/titanic/train.csv')
# Data preprocessing
train = train[train['Pclass'] == 3]
train = train.drop(['Name', 'Ticket', 'Cabin', 'PassengerId'], axis=1)
train.Sex = train.Sex.map({'female': 0, 'male': 1})
median_age_men = train[train['Sex'] == 1]['Age'].median()
median_age_women = train[train['Sex'] == 0]['Age'].median()
train.loc[train.Age.isnull() & (train['Sex'] == 1), 'Age'] = median_age_men
train.loc[train.Age.isnull() & (train['Sex'] == 0), 'Age'] = median_age_women
train.Embarked = train.Embarked.map({'S': 0, 'C': 1, 'Q': 2, 'nan': 'NaN'})
train = train.dropna(inplace=False)
```

Mark only one oval.

	1	2	3	4	5	
Leas	◯	◯	◯	◯	◯	Most representative

24. Scenario 2: Script 4 helpfulness

Mark only one oval.

	1	2	3	4	5	
Leas	◯	◯	◯	◯	◯	Most helpful

25. Scenario 2: Without being given the most useful script above, how much time do you think you need to confidently compose an equivalent script **independently on your own**?

Mark only one oval.

- ◯ Less than 1 hour
- ◯ 1-2 hours
- ◯ 2-3 hours
- ◯ More than 3 hours

26. Scenario 2: Without being given the script you chose, but if you have access to the collection of 60 historical scripts, how much time do you think you need to confidently compose an equivalent script, **informed by the historical scripts**?

Mark only one oval.

- ◯ Less than 1 hour
- ◯ 1-2 hours
- ◯ 2-3 hours
- ◯ More than 3 hours