COURSE HANDOUT · 4TH YEAR SOFTWARE ENGINEERING

Data Mining

Practical Labs (TP)

Chapter 2

Data Preprocessing

PRESENTED BY	Dr. Sihem BENFRIHA
DEPARTMENT	Computer Sciences — University of Tlemcen
CONTACT	benfriha.sihem@univ-tlemcen.dz · benfriha.sihem@gmail.com

VERSION	4.0
DATE	22/04/2026

Table of Contents

11. Chapter 2 : Data Preprocessing/ Duration 4h 30 min

11.1. objectives

- **Identify** common data preprocessing techniques (cleaning, normalization, encoding, feature selection).
- **Explain** the importance of data preprocessing in the machine learning pipeline.
- **Use** Python libraries to clean and transform datasets (handle missing values, normalize data, encode categorical variables).
- **Examine** datasets to detect issues (outliers, missing values, inconsistencies).
- **Combine** multiple preprocessing steps into a complete data pipeline.
- **Assess** the impact of preprocessing techniques on model performance.

11.2. Prerequisites

- Basic Python Programming.
- Fundamental Mathematics & Statistics.
- Logical and Analytical Thinking

11.3. TP 3 : session 1

Task 1 import the basic libraries

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
sns.set() # Will import Seaborn functionalities
# we don't like warnings
# you can comment the following 2 lines if you'd like to
import warnings
warnings.filterwarnings('ignore')
```

Task 2 Load the Data

1. Read the CSV file into a DataFrame named autinsurance:

```
autinsurance = pd.read_csv('insurance_claimsV3.csv')
autinsurance.head()
```

2. Display:

- The first 5 rows (head()).
- The shape of the dataset (.shape).
- The column names (.columns).

Task 3 Distribution of target variable

1. Use autinsurance.info() to inspect data types and non-null counts.
2. Use autinsurance.describe() on numerical columns.
3. Compute the distribution of the target variable:

autinsurance['fraud_reported'].value_counts()

4. To calculate fractions, pass `normalize=True` to the `value_counts` function.

autinsurance['fraud_reported'].value_counts(normalize=True)

5. Plot the distribution of the target variable as a bar plot.

(autinsurance['fraud_reported'].value_counts().plot(

kind='bar',

figsize=(8, 6),

title='Distribution of Target Variable',

)

);

plt.show()

Task 4. Handling Missing Values

Detect Missing Values

1. Start by checking missing values:

Disply all Columns

pd.options.display.max_rows=170

Finding missing values

autinsurance.isna().sum()

2. Delete rows with NaN values Once you have figured out all the missing details, we remove all the missing rows from the DataFrame. we use the dropna() function:

removing Null values

autinsurance = autinsurance.dropna()

autinsurance.info()

Task 5 Outlier Detection and Treatment

Visualize Outliers with Boxplots

1. Select 2–3 numerical features (e.g. total_claim_amount, injury_claim, etc.).

2. Plot boxplots using either pandas or seaborn:

plt.figure(figsize=(8, 6))

sns.boxplot(x=autinsurance["total_claim_amount"])

```
plt.show()

# Boxplots

# Horizontal boxplot with observations

plt.rcParams['figure.figsize'] = 8,4

sns.boxplot(autinsurance['months_as_customer'])
```

[cf. TP3]

11.4. TP 3 : session 2

Définition : Feature Selection

Collinear Features : are features that are highly correlated with one another. In machine learning, these lead to decreased generalization performance on the test set due to high variance and less model interpretability.

The identify_collinear method finds collinear features based on a specified correlation coefficient value. For each pair of correlated features, it identifies one of the features for removal (since we only need to remove one):

Méthode

1. Create a correlation matrix for numerical features:

```
# Create correlation matrix

au_copy = autinsurance.copy()

corr_matrix = au_copy.corr().abs()

# Select upper triangle of correlation matrix

upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))

# Find index of feature columns with correlation greater than 0.7

to_drop = [column for column in upper.columns if any(upper[column] > 0.7)]
```

2. Which features are strongly correlated with each other?

3. Remove features that have a correlation > 0.7 with another feature (highly correlated pairs).

4. Which features did you decide to drop due to multicollinearity?

```
# Correlation with Target Variable

corr_matrix = au_copy.corr().abs()

plt.figure(figsize=(30,20))

au_copy.corr()['fraud_reported'].plot()

#autinsurance.corr()[:].plot()

#autinsurance.corr().plot()
```

Model-Based Feature Selection

1. Encode categorical variables (e.g. using pd.get_dummies) to create a numeric dataset.

```
autinsurance_v3 = au_copy.copy()
```

autinsurance_v3.shape

print(autinsurance_v3.columns)

2. Split the dataset into features X and target y:

X = autinsurance_v3.drop("fraud_reported", axis=1)

y = autinsurance_v3["fraud_reported"]

X.shape

y.shape

3. Splitting the dataset into the Training set and Test set

Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split

training_features, test_features, \

training_target, test_target, = train_test_split(autinsurance_v3.drop('fraud_reported', axis = 1),

autinsurance_v3.fraud_reported,

test_size = .2,

random_state=12)

4. Train a simple model such as XGboost inspect feature importances or coefficients.

from xgboost import XGBClassifier

from matplotlib import pyplot

plt.figure(figsize=(20,10))

fit model no training data

model = XGBClassifier()

model.fit(training_features, training_target)

feature importance

#print(model.feature_importances_)

plot

pyplot.bar(range(len(model.feature_importances_)), model.feature_importances_)

pyplot.show()

5. Which features are most important according to the model?

feature_importances = pd.DataFrame({'Importance Coef' :model.feature_importances_ , 'Features' : training_features.columns})

feature_importances.nlargest(135, 'Importance Coef')

6. Save the Prepared Dataset

1. After all preprocessing steps (missing values, outliers, feature selection), create a final version of the dataset, e.g.:

autinsurance_clean = autinsurance.copy()

2. Save it to disk:

Save to file

autinsurance_v3.to_csv("insurance_claimsV4.csv")

3. What is the final number of features (columns) and rows?

autinsurance_v3.shape

11.5. Final Test

Exercice : self evaluation Exercice

Exercice : Exercice 1 Loading and Exploring Data (Basic)

- Load a dataset using pandas (e.g., CSV file)
- Display:
- First 5 rows
- Data types of each column
- Basic statistics (describe())

Exercice : Exercice 2 Handling Missing Values

Given a dataset with missing values:

- Identify missing values using .isnull()
- Replace missing values:
- Numerical columns → mean
- Categorical columns → most frequent value

Exercice : Exercice 3 Data Cleaning

Remove duplicate rows

Standardize column names (lowercase, no spaces)

Convert a column to the correct data type (e.g., string → integer)