FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

SEMESTER 2 SESSION 2024/2025

CSC4600

Dr. Raihani

Supervised Learning: Python Programming

NAME	MATRIC
MOHAMMAD JAVAN SAMBOEPUTRA HERLAMBANG	213878

1. Introduction

This project explores supervised learning techniques using two real-world datasets: the Titanic passenger data (for binary classification) and a House Price dataset (for regression). The objective is to demonstrate a complete machine learning workflow—from data preprocessing and exploratory data analysis (EDA) to model training, evaluation, and comparison. For each dataset, two algorithms are applied to assess their performance on relevant metrics. The Titanic dataset uses Logistic Regression and Random Forest Classifier to predict survival, while the House dataset uses Linear Regression and Random Forest Regressor to predict housing value. The goal is to compare model effectiveness, identify key patterns in the data, and draw insights from predictive performance.

2. Dataset Overview

Titanic Dataset

- Source: Kaggle
- Target Variable: *Survived* (1 = survived, 0 = died)
- Features: Pclass, Sex, Age, SibSp, Parch, Fare, Embarked

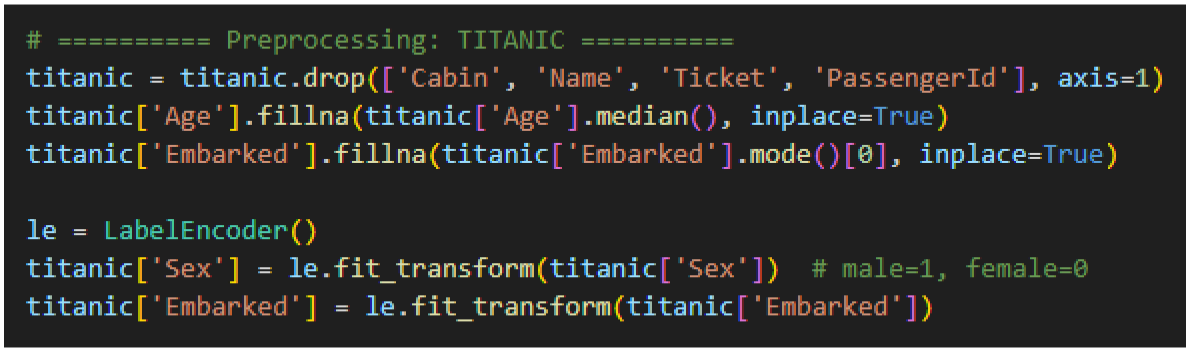
House Price Dataset

- Source: -
- Target Variable: *Value* (house price)
- Features: Rooms, Distance

3. Data Preprocessing

Titanic Dataset

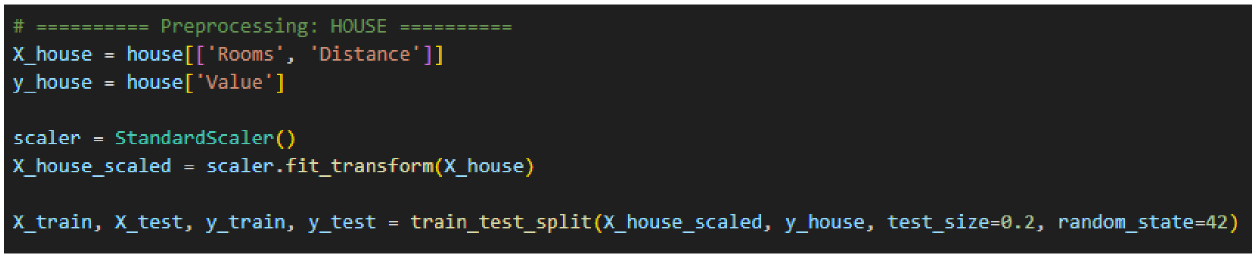
- Dropped columns: Cabin, Name, Ticket, PassengerId
- Filled missing values in Age (median), Embarked (mode)
- Encoded Sex and Embarked with Label Encoding



```python
# ========== Preprocessing: TITANIC ==========
titanic = titanic.drop(['Cabin', 'Name', 'Ticket', 'PassengerId'], axis=1)
titanic['Age'].fillna(titanic['Age'].median(), inplace=True)
titanic['Embarked'].fillna(titanic['Embarked'].mode()[0], inplace=True)

le = LabelEncoder()
titanic['Sex'] = le.fit_transform(titanic['Sex'])  # male=1, female=0
titanic['Embarked'] = le.fit_transform(titanic['Embarked'])
```

House Price Dataset

- No missing values
- Scaled *Rooms* and *Distance* using *StandardScaler*

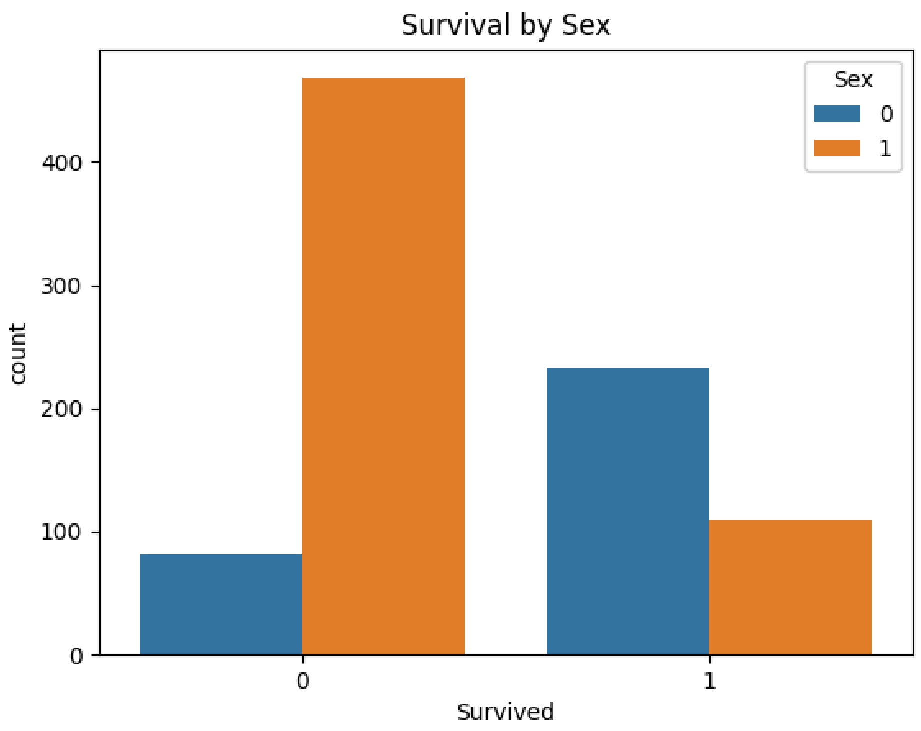


```python
# ========== Preprocessing: HOUSE ==========
X_house = house[['Rooms', 'Distance']]
y_house = house['Value']

scaler = StandardScaler()
X_house_scaled = scaler.fit_transform(X_house)

X_train, X_test, y_train, y_test = train_test_split(X_house_scaled, y_house, test_size=0.2, random_state=42)
```

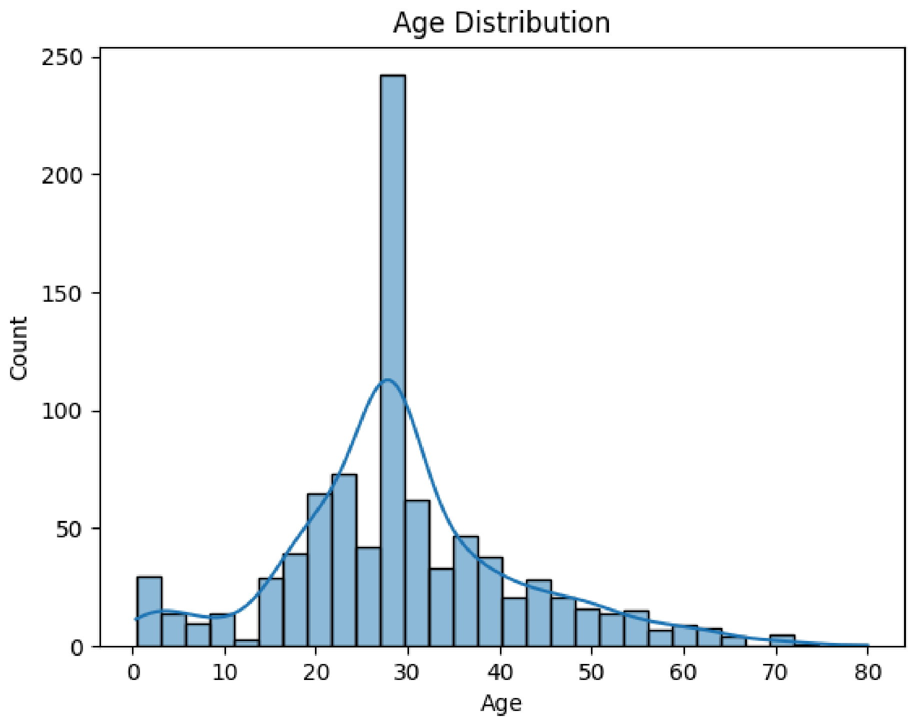
4. Exploratory Data Analysis (EDA)

4.1 Titanic Dataset

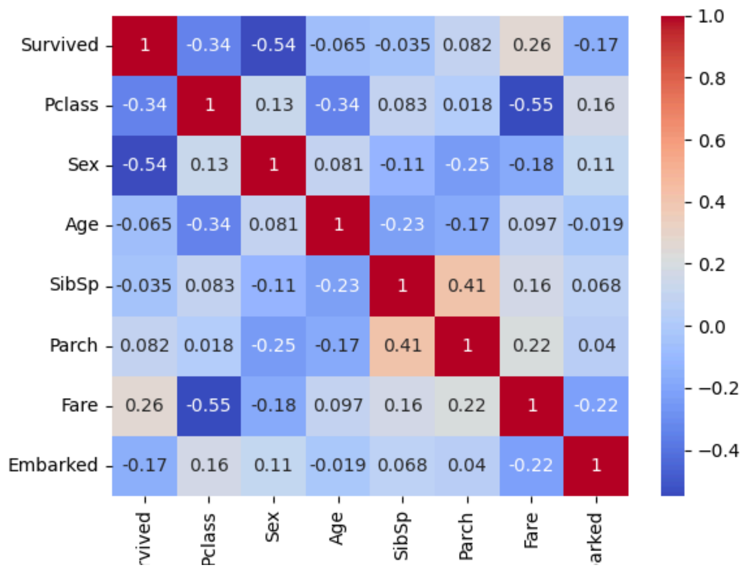
- Survival count by sex (Bar chart)



- Age distribution (Histogram with KDE)



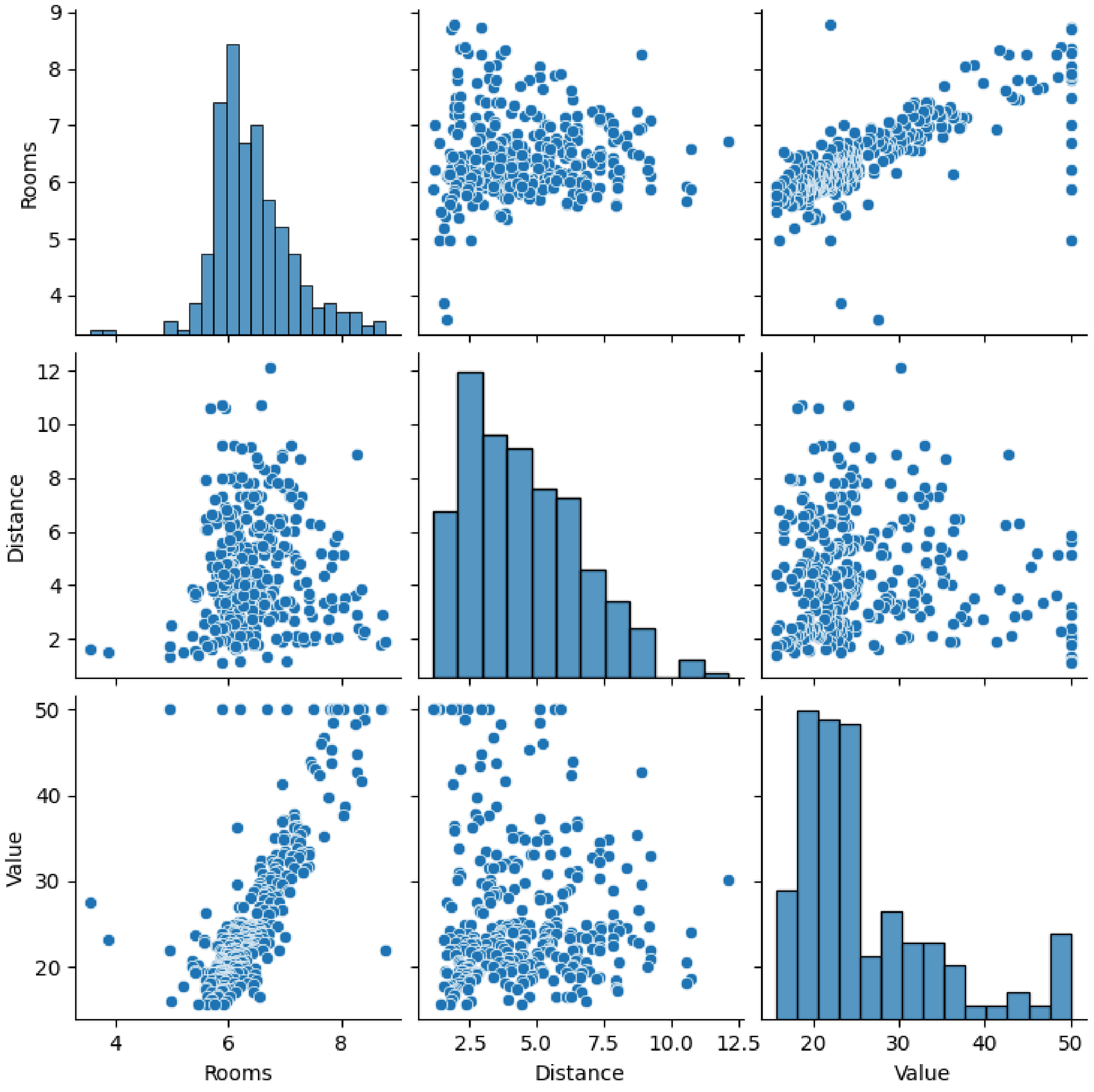
- Correlation heatmap



Key Findings:

- Females had higher survival rates
- First-class passengers survived more
- Age is approximately normally distributed

4.2 House Price Dataset



📝 Key Findings:

- *Rooms* and *Distance* have a visible influence on house *Value*
- *Rooms* is positively correlated; *Distance* is negatively correlated

5. Model Building and Results

Titanic Dataset (Classification)

- Models Used: Logistic Regression, Random Forest Classifier
- Metric Used: F1 Score

Model	F1 Score
Logistic Regression	0.764
Random Forest	0.775

Random Forest outperforms Logistic Regression slightly, likely due to its ability to handle non-linear patterns.

House Price Dataset (Regression)

- Models Used: Linear Regression, Random Forest Regressor
- Metrics Used: RMSE, R^2 Score

Model	RMSE	R^2 Score

Linear Regression	3.994	0.768
Random Forest	3.383	0.833

Random Forest provides better performance due to its robustness to non-linearity and outliers.

6. Conclusion

This project demonstrates the practical application of supervised learning on both classification and regression tasks using Python. In both datasets, Random Forest models outperformed their simpler linear counterparts - highlighting the advantages of ensemble methods in capturing complex, non-linear relationships in real-world data. Proper data preprocessing (e.g., handling missing values, feature encoding, and scaling) was critical in preparing the datasets for modeling. Exploratory analysis also revealed valuable insights, such as the survival advantage of certain groups in the Titanic dataset and the influence of room count and distance on house prices. Overall, this project reinforces the importance of end-to-end data handling and model evaluation in achieving reliable predictions.

Appendix

1. Code

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.metrics import accuracy_score, f1_score, mean_squared_error,
    r2_score


titanic = pd.read_csv("titanic.csv")

house = pd.read_csv("house_prices.csv")


# ========== Preprocessing: TITANIC ==========

titanic = titanic.drop(['Cabin', 'Name', 'Ticket', 'PassengerId'], axis=1)

titanic['Age'].fillna(titanic['Age'].median(), inplace=True)

titanic['Embarked'].fillna(titanic['Embarked'].mode()[0], inplace=True)


le = LabelEncoder()

titanic['Sex'] = le.fit_transform(titanic['Sex'])  # male=1, female=0

titanic['Embarked'] = le.fit_transform(titanic['Embarked'])


# ========== TERMINAL-BASED EDA: TITANIC ==========

print("\n===== TITANIC DATASET EDA =====")


print("\n[1] Survival Count:")

print(titanic['Survived'].value_counts())


print("\n[2] Survival Rate by Sex:")

print(titanic.groupby('Sex')['Survived'].mean())
print("\n[3] Survival Rate by Pclass:")

print(titanic.groupby('Pclass')['Survived'].mean())


print("\n[4] Statistical Summary:")

print(titanic.describe())


print("\n[5] Correlation with Survival:")

print(titanic.corr(numeric_only=True)['Survived'].sort_values(ascending=False))


# ========== PLOTS: TITANIC ==========

sns.countplot(x='Survived', hue='Sex', data=titanic)

plt.title('Survival by Sex')

plt.show()


sns.histplot(titanic['Age'], kde=True)

plt.title('Age Distribution')

plt.show()


sns.heatmap(titanic.corr(), annot=True, cmap='coolwarm')

plt.title('Titanic Correlation Heatmap')

plt.show()
```

```
# ========== MODELING: TITANIC ==========

from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier


X = titanic.drop('Survived', axis=1)

y = titanic['Survived']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)


logreg = LogisticRegression(max_iter=1000)

rf = RandomForestClassifier(random_state=42)


logreg.fit(X_train, y_train)

rf.fit(X_train, y_train)


y_pred_log = logreg.predict(X_test)

y_pred_rf = rf.predict(X_test)


print("\nLogistic Regression F1:", f1_score(y_test, y_pred_log))

print("Random Forest F1:", f1_score(y_test, y_pred_rf))



# ========== Preprocessing: HOUSE ==========
X_house = house[['Rooms', 'Distance']]

y_house = house['Value']


scaler = StandardScaler()

X_house_scaled = scaler.fit_transform(X_house)


X_train, X_test, y_train, y_test = train_test_split(X_house_scaled, y_house,
    test_size=0.2, random_state=42)


# ========== TERMINAL-BASED EDA: HOUSE PRICE ==========

print("\n===== HOUSE PRICE DATASET EDA =====")


print("\n[1] Statistical Summary:")

print(house.describe())


print("\n[2] Correlation with Value:")

print(house.corr()['Value'].sort_values(ascending=False))


print("\n[3] Distribution of Value (Grouped):")

print(pd.cut(house['Value'], bins=5).value_counts())


# ========== PLOTS: HOUSE ==========

sns.pairplot(house)

plt.show()
```

```
sns.heatmap(house.corr(), annot=True, cmap='coolwarm')

plt.title('House Price Correlation Heatmap')

plt.show()


# ========== MODELING: HOUSE PRICE ==========

from sklearn.linear_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor


lr = LinearRegression()

rf_reg = RandomForestRegressor(random_state=42)


lr.fit(X_train, y_train)

rf_reg.fit(X_train, y_train)


y_pred_lr = lr.predict(X_test)

y_pred_rf = rf_reg.predict(X_test)


print("\nLinear Regression RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_lr)))

print("Random Forest RMSE:", np.sqrt(mean_squared_error(y_test, y_pred_rf)))


print("\nLinear Regression R2:", r2_score(y_test, y_pred_lr))

print("Random Forest R2:", r2_score(y_test, y_pred_rf))
```