IJSES
Volume 9, Issue 5, pp. 120-127, 2025.
International Journal of Scientific Engineering and Science
ISSN (Online): 2456-7361

The Application of Python to Business Data Analysis using Basic Machine Learning Models

Hoang Thi Thu Ha
Department of Digital Economy, Faculty of Mathematical Economics, Thuongmai University
Email address: hoangha.math@tmu.edu.vn

Abstract— *This study aims to equip students with a foundational understanding of the business data analysis process. By utilizing Python and widely used libraries such as pandas, scikit-learn, matplotlib, and seaborn, it demonstrates key steps including data preprocessing, exploration, feature transformation, and the development and evaluation of machine learning models. The dataset, obtained from Kaggle, comprises 200,000 records related to marketing campaigns with variables such as campaign type, customer segment, cost, and impressions. Through the implementation of regression models—Linear Regression, Decision Tree, and Random Forest—the study helps learners grasp practical data analysis workflows and strengthens their ability to apply Python to real-world business challenges.*

Keywords— *Business data analysis, Python, linear regression, decision tree, random forest.*

I. Introduction

In the context of the rapid development of the digital economy, data-driven decision-making has become an inevitable trend for modern businesses. Business data analysis not only enables companies to better understand customer behavior and market trends but also provides a scientific foundation for optimizing operations, refining marketing strategies, and forecasting business outcomes. As a result, equipping students with knowledge and skills in data analysis has become a vital component of training programs in economics, management, and finance.

Currently, machine learning models are increasingly widely applied in data analysis due to their ability to uncover hidden patterns and provide highly accurate forecasts. Among these, many basic and accessible models that still deliver significant analytical efficiency are often preferred for business and research problems. However, selecting the appropriate model depends on the analysis objectives as well as the characteristics of the dataset.

Specifically, the Linear Regression model is frequently used to forecast continuous variables such as revenue, cost, or profit, based on a linear relationship between dependent and independent variables. The Decision Tree model is commonly applied to classification or regression tasks, offering advantages such as ease of interpretation and intuitiveness, making it especially suitable for scenarios where understanding the decision-making logic is crucial. Meanwhile, the Random Forest model is a powerful machine learning technique that combines multiple decision trees to improve accuracy and reduce overfitting, leveraging information from multiple subsamples.

To effectively implement machine learning models in business data analysis, Python has established itself as one of the most popular and powerful programming tools today. With its simple and accessible syntax, along with a rich ecosystem of libraries such as scikit-learn, pandas, and matplotlib, Python enables users to quickly build, train, and evaluate machine learning models. Additionally, a large user community and diverse learning resources have helped promote Python as the preferred choice in data science and artificial intelligence. Currently, Python is widely taught in university programs worldwide as a foundational programming language for data science and machine learning.

This article aims to provide a detailed hands-on guide for college students, helping them engage with the process of analyzing business data in Python using three basic machine learning models: linear regression, decision trees, and random forests. By illustrating real data and offering step-by-step instructions, the article not only enhances students' data processing and analytical skills but also fosters quantitative thinking—an essential competency in the era of data-driven business.

II. Theoretical Framework And Literature Review

2.1. Theoretical framework

All three models - Linear Regression, Decision Tree Regressor, and Random Forest Regressor - belong to the category of supervised learning algorithms. However, each model has its own fundamental concept, operating mechanism, as well as distinct advantages and disadvantages.

Linear Regression

Linear regression is a statistical method used to model the linear relationship between a dependent variable Y and one or more independent variables, with the goal of predicting the dependent variable's outcome. The multivariate linear regression model is expressed by the formula: $Y, X_1, X_2, \dots, X_k$

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k + U, \quad (1)$$

In this model, β_0 is the intercept (also known as the blocking coefficient), and β_j is the slope (angular coefficient), representing the degree and direction of the impact of the variable X_j on the dependent variable Y, for $(j = 1, 2, \dots, k)$; U denotes the random error term.

In the sample, these coefficients are estimated and referred to as sample regression coefficients, denoted as $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_k$, which serve as estimates of the population

IJSES
Volume 9, Issue 5, pp. 120-127, 2025.
International Journal of Scientific Engineering and Science
ISSN (Online): 2456-7361

regression coefficients $\beta_0, \beta_1, \ldots, \beta_k$. Therefore, the estimated regression model is expressed as:

$$Y = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_k X_k + \varepsilon \quad (2)$$

with the so-called surplus, ε, which is the estimate of U

One of the most common methods for estimating model (2) is the Ordinary Least Squares (OLS) method. The regression coefficients are calculated in this manner by minimizing the sum of the squared errors (residuals). However, to apply OLS effectively, model (1) must satisfy several basic assumptions, including: a linear relationship between the dependent and independent variables; uncorrelated random errors; normally distributed random errors; constant variance of the errors (homoscedasticity); and no severe multicollinearity among the independent variables.

Decision Tree Regressor

The Decision Tree Regressor is a supervised machine learning model used to predict continuous values in regression problems. It works by partitioning the feature space into regions based on decision rules and assigning a prediction value—typically the average of the data points in each region—to each region. The model constructs a tree structure where:

- the root node represents the entire dataset,
- internal nodes apply conditions based on feature values (e.g., "marketing costs > $1000"),
- leaf nodes contain the predicted value (usually the average or median of the data points in that region).

One of the core principles in building a decision tree is selecting the attribute for data partitioning that maximizes the ability to distinguish between classes. To achieve this, two important concepts are used: Entropy and Information Gain.

Entropy measures the degree of heterogeneity (or disorder) in a dataset and was introduced in Shannon's (1948) information theory. It reflects the uncertainty of information in a random variable. For a dataset with classification classes, entropy is calculated using the following formula:

$$Entropy(S) = -\sum_{i=1}^{n} p_i . \ln(p_i)$$

In which, is p_i the probability of the appearance of the class in the dataset S. When all the elements in belong to the same class, the entropy reaches a value of 0. In contrast, entropy reaches its maximum value when the layers are evenly distributed, signifying maximum uncertainty.S

Information Gain (IG) is calculated through Entropy. This metric indicates how much entropy is reduced when dividing the dataset by a specific attribute. The IG of the attribute for the dataset is defined as follows:Type equation here.

$$IG(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} . Entropy(S_v)$$

in which:

- $Values(A)$ is the set of possible values of attribute A,
- S_v is a subset of containing valuable elements,$SA = v$
- $|S|, |S_v|$ is the size of the corresponding set.S, S_v

Information Gain (IG) measures the improvement in the "purity" of the data after a split. In the ID3 algorithm, the attribute with the highest Information Gain is selected to partition the dataset at each node, optimizing the classification performance of the decision tree.

Decision tree models offer several significant advantages, including easy and intuitive interpretation when represented as a tree, efficient handling of nonlinear relationships between input and output variables, and no requirement for data normalization or linearity assumptions. However, the model also has limitations. If not carefully fine-tuned, the tree can grow too deep, leading to overfitting. Additionally, compared to ensemble models such as Random Forest or Gradient Boosting, the accuracy of a single decision tree is generally lower and more sensitive to noise in the data.

In practice, decision tree models are widely applied to various forecasting problems, such as predicting sales, marketing costs, or marketing campaign performance indicators, based on input features like advertising spend, number of customers, or communication channels used.

Random Forest Regressor

The Random Forest Regressor is a supervised machine learning model commonly used for regression tasks to predict continuous values. It is an ensemble method based on the bagging technique (Bootstrap Aggregating), which combines multiple independently trained decision trees built on randomly selected subsets of data and features. The final prediction is obtained by averaging the outputs from all individual trees.

One of the key advantages of the Random Forest is its ability to efficiently handle nonlinear relationships and complex data, while reducing the risk of overfitting through its ensemble learning mechanism and averaging of results. The model also provides insights into feature importance, which aids data analysis and interpretation.

However, Random Forest has some limitations. Compared to simpler models like linear regression or single decision trees, it requires more computational resources and is harder to interpret due to its complexity. In certain cases—especially with very large datasets or when extremely high accuracy is needed—deep learning models may prove more effective.

In practice, the Random Forest Regressor is widely used in forecasting tasks such as predicting sales, click-through rates (CTR), conversion rates, and other marketing performance indicators based on features like advertising costs, media channels used, and consumer behavior.

2.2. Literature review

In recent years, alongside the rapid development of machine learning, there has been a growing number of empirical studies applying these models to business data analysis. A notable example is the study by Zhou et al. (2021), which used linear regression analysis to evaluate the effectiveness of YouTube advertising on sales revenue. The dataset included a company's advertising budget and sales figures, sourced from the Business of Apps website, with advertising costs ranging from $8.79 to $352.75. The entire analysis was conducted using R. Similarly, Takale et al. (2022) applied a linear regression model to predict sales revenue based on advertising costs. Their study utilized historical data from a public advertising dataset on the Kaggle platform, including variables such as advertising expenses on TV, radio, newspapers, and corresponding revenue. The

analysis and model development were performed entirely using Python, supported by popular libraries such as scikit-learn, pandas, and matplotlib.

Meanwhile, Gkikas and Theodoridis (2024) applied a decision tree model to classify users into three interaction levels: low, medium, and high. The study utilized data collected from Google Analytics of an online fashion retailer in Greece over a six-month period (June 26, 2023, to December 25, 2023). The metrics included sessions, number of events, purchase revenue, number of transactions, conversion rate, and bounce rate. The entire analysis was conducted using Python, supported by libraries such as scikit-learn, pandas, matplotlib, seaborn, and NumPy. This research provides an empirical foundation for optimizing digital marketing strategies through user behavior analysis.

The Random Forest model is also commonly used in business analysis. For example, Pes (2021) applied this model to predict high-value customer segments in the e-commerce context. The data, collected from an online retailer's CRM system between 2018 and 2020, included purchase history, average order value, shopping frequency, and demographic characteristics such as age, gender, and region. The study not only focused on building predictive models but also addressed the issue of imbalanced data using sample balancing techniques and evaluated the importance of features influencing customer segmentation. The entire process was conducted in Python with support from libraries such as scikit-learn, pandas, matplotlib, and seaborn.

III. Data And Methodology

3.1 Data

The illustrative data for the Python commands in this study were sourced from the Kaggle platform (https://www.kaggle.com/). The dataset contains 200,000 observations of marketing campaigns with 15 variables. For this analysis, the author uses the output variable 'Conversion_Rate' and the input variables: 'Campaign_Type', 'Target_Audience', 'Duration', 'Channel_Used', 'Acquisition_Cost', 'Impressions', and 'Customer_Segment'. The variables are described in Table 1.

Table 1. List of variables used in the analysis

Variables	Description	Type
Conversion Rate	The percentage of leads or impressions that converted into desired actions, indicating campaign effectiveness.	Float
Campaign Type	The type of campaign employed, including email, social media, influencer, display, or search.	Object
Target Audience	The specific audience segment targeted by the campaign, such as women aged 25-34, men aged 18-24, or all age groups.	object
Duration	The duration of the campaign, expressed in days.	int
Channels Used	The channels utilized to promote the campaign, which may include email, social media platforms, YouTube, websites, or Google Ads.	object
Acquisition Cost	The cost incurred by the company to acquire customers, presented in monetary format.	int
Impressions	The total number of times the campaign was displayed or viewed by the target audience.	int
Customer Segment	The specific customer segment or audience category that the campaign was tailored for, such as tech enthusiasts, fashionistas, health and wellness enthusiasts, foodies, or outdoor adventurers.	object

Source: https://www.kaggle.com/

3.2 Methodology

This paper employs a practical research approach using simulated data from Kaggle to demonstrate the process of applying Python in business data analysis with three fundamental machine learning models: Linear Regression, Decision Tree, and Random Forest.

The research process encompasses key steps including data preprocessing, exploratory data analysis, and the development and evaluation of machine learning models. The selected dataset includes input variables related to marketing campaigns and the output variable, Conversion Rate.

The models are implemented and assessed using popular Python libraries such as pandas, scikit-learn, and matplotlib, ensuring the approach is practical, easy to understand, and suitable for college students engaged in learning and research.

IV. Research Results

The following presents the business data analysis process carried out using Python software.

4.1. Import Required Libraries

To perform data analysis, we first need to import the necessary libraries. These libraries provide tools for data processing, machine learning model development, result visualization, and model performance evaluation.

```
!pip install python-docx
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.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import warnings
warnings.filterwarnings('ignore')
```

After importing the required libraries, the next step is to load the data into the working environment for processing and analysis.

4.2. Reading Data

In this study, the data is stored on Google Drive. Therefore, the first step is to connect Google Drive to the working environment in Google Colab. Once the connection is established, the data file can be read from the specified path for use in the analysis process.

```
from google.colab import drive
drive.mount('/content/drive')
df = pd.read_csv('/content/drive/My Drive/Journal/marketing_campaign_dataset.csv')
```

4.3. Data Discovery

The purpose of this step is to provide an initial overview of the dataset to better understand the characteristics of each variable, the data structure, and potential anomalies. Specifically, the data exploration includes:
(1) Looking at summary data such the number of variables, the data types in each column, and the count of missing values (if any).

```
print(df.info())
```

(2) Descriptive statistics for quantitative variables include the mean, standard deviation, minimum and maximum values, as well as percentiles.

```
# Variable Separation
df.describe()
# Separation of classification variables
df.describe(include='object')

from docx import Document
doc = Document()
# Variable description statistics
doc.add_heading("Statistics describing variables", level=1)
desc_num = df.describe()
doc.add_paragraph(desc_num.to_string())
# Classification variable description statistics
doc.add_heading("Statistics describing taxonomic variables", level=1)
desc_cat = df.describe(include='object')
doc.add_paragraph(desc_cat.to_string())

doc.save("/content/drive/My Drive/Journal/thong_ke_mo_ta.docx")
```

(3) Detect anomalies in the data—such as outliers, misformatted entries, and missing values—to develop appropriate strategies for the subsequent preprocessing step.

```
# Missing Data Detection
df.isnull().sum()
# Foreign Data Detection
df.hist(figsize=(12,8))
plt.tight_layout()
plt.show()
```

This step is essential to ensure the quality of input data for machine learning models and to support the selection of appropriate variables, which will guide subsequent data processing and analysis.

4.4. Data pre-processing

If, during the data discovery step, missing values or extraneous observations are found, they need to be properly handled.

```
# Handling Missing Observations
df = df.dropna()
# Exotic Processing by IQR
for col in ['Conversion_Rate', 'Acquisition_Cost', 'Impressions', 'Duration']:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    df = df[(df[col] >= Q1 - 1.5 * IQR) & (df[col] <= Q3 + 1.5 * IQR)]
```

In addition, to ensure that the data is fed into the model in the correct format and scale, it is sometimes necessary to perform two important operations: standardizing the variables and encoding categorical variables. Quantitative variables often have different units of measurement and value scales (e.g., costs in hundreds of dollars, time in days, percentages, etc.). Without standardization, machine learning models such as linear regression or k-NN may be biased toward variables with larger values, affecting prediction accuracy. Therefore, normalization helps bring variables to the same scale, usually with a mean of 0 and a standard deviation of 1 (z-score normalization), making the learning model more effective. Additionally, machine learning models cannot directly process categorical data (such as campaign names, communication channels, customer segments). Thus, categorical variables need to be encoded into numerical form. Common methods include one-hot encoding (creating binary columns for each category, suitable for variables with a limited number of levels) and label encoding (assigning numeric values to categories, used when the variable has a natural order).

```
from sklearn.preprocessing import StandardScaler, LabelEncoder
# Normalize variables
scaler = StandardScaler()
df[['Conversion_Rate', 'Acquisition_Cost', 'Impressions', 'Duration']] = scaler.fit_transform(
df[['Conversion_Rate', 'Acquisition_Cost', 'Impressions', 'Duration']])

# Encoding of grouping variables
label_enc = LabelEncoder()
for col in ['Campaign_Type', 'Target_Audience', 'Channel_Used', 'Customer_Segment']:
df[col] = label_enc.fit_transform(df[col])

# Display descriptive information for all processed variables
print("\nDescription information of normalized and encoded variables:")
print(df[['Conversion_Rate', 'Acquisition_Cost', 'Impressions', 'Duration',
'Campaign_Type', 'Target_Audience', 'Channel_Used', 'Customer_Segment']].describe())

#Hiển more detailed information for each variable
for col in ['Conversion_Rate', 'Acquisition_Cost', 'Impressions', 'Duration',
```

International Journal of Scientific Engineering and Science
Volume 9, Issue 5, pp. 120-127, 2025. ISSN (Online): 2456-7361

```
'Campaign_Type', 'Target_Audience', 'Channel_Used', 'Customer_Segment']:
print(f"\nStats for {col}:")
print(df[col].describe())
```

4.5. Identify linear relationships between variables

In data analysis and predictive model building, understanding the relationships between input and output variables plays a crucial role in feature selection, result interpretation, and improving model performance. One useful tool for this purpose is the correlation matrix. The goal of this step is to determine whether there are strong or weak linear relationships among independent variables as well as between independent variables and the dependent variable (target variable—in this case, *Conversion_Rate*). If two input variables are highly correlated (multi-collinear), one of them should be removed to avoid redundancy. At the same time, priority should be given to including variables that show a strong correlation with the target variable when building the forecasting model. A popular visualization tool for this is a heatmap, which helps analysts easily identify relationships between variables through color coding.

```
# – Correlation Matrix:

cols_to_drop = ['Company', 'Location', 'Language', 'Date', 'dtype']
cols_to_drop_existing = [col for col in cols_to_drop if col in df.columns]

if cols_to_drop_existing:
df_numeric = df.drop(columns=cols_to_drop_existing)
else:
df_numeric = df.copy() # If no columns to drop, work on a copy

corr = df_numeric.corr()

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

4.6. Feature transformation and feature engineering

In the process of data analysis and predictive model building, feature transformation and feature engineering are essential steps to improve model performance and enhance data interpretability. Specifically:

- Logarithmic transform

Applying logarithms to variables such as *impressions* helps reduce the influence of outliers by smoothing right-skewed distributions, making them closer to normal distribution. This aligns better with the assumptions of regression models and can also help linearize relationships between input and output variables.

- Inverse transformation

This transformation is often applied when a variable exhibits a nonlinear relationship with the target variable. Taking the inverse highlights inverse relationships and also reduces the impact of large values, similar to the logarithmic transform, but it suits some specific types of distributions better.

- Interaction features

Creating new features by combining two or more input variables can capture complex relationships that simple linear models might miss. Moreover, interaction features improve model predictability when variables jointly affect the target variable.

```
# Get logarithms
df['log_Impressions'] = np.log1p(df['Impressions'])
# Take the inverse
df['inv_Duration'] = 1 / (df['Duration'] + 1e-5)
# Create Interactive Variables
df['Cost_Impressions'] = df['Acquisition_Cost'] * df['Impressions']
```

7. Split the dataset into trains and test

An important step in creating machine learning models is dividing the dataset into training and testing sets. The training set is used to "teach" the model to identify connections in the data, which enables the model to fine-tune its parameters to best match the data's features. Meanwhile, the testing set evaluates the model's performance on new, unseen data, thereby measuring its generalization and predictive ability in practice.

Additionally, this division helps to avoid overfitting, which is a situation in which the model fits the training data too well but does not do well on fresh data. By splitting the data into training and testing subsets, we ensure that evaluation results more accurately reflect the model's true effectiveness. Typically, the common split ratio is 70:30 or 80:20. In this step, the data is randomly divided to maintain representativeness in both sets.

```
# Definition of input variable y and output variable X
X = df.drop('Conversion_Rate', axis=1)
y = df['Conversion_Rate']
# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

8. Some machine learning models used in business data analysis

In this section, we will present the application of three machine learning models—linear regression, decision trees, and random forests—to analyze business data. The process consists of three main steps:

Step 1. Model Training

Create the model using the training dataset so that it may learn key correlations and trends in the data.

Step 2. Model Evaluation

Evaluate the model's performance on the test dataset to assess its predictability and generalization ability. Common evaluation metrics include accuracy, mean squared error, or confusion matrix, depending on the type of problem.

Step 3. Visualize the Results

Display the analysis outcomes, predictions, and model performance through charts and graphs. Visualization helps

identify trends, compare results effectively, and communicate insights clearly.

8.1 Linear Regression Model

Step 1. Model Training

```
from sklearn.linear_model import LinearRegression

# Initialization and training of linear regression models
model = LinearRegression()
model.fit(X_train, y_train)

# Extraction of blocking and regression coefficients
intercept = model.intercept_
Coefficients = model.coef_

print("\nIntercept:", intercept)
print("Coefficients:")
for feature, coef in zip(X.columns, coefficients):
    print(f"{feature}: {coef:.4f}")
```

Step 2. Model Evaluation

```
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score
import numpy as np

# Predict the output value on the test set
y_pred = model.predict(X_test)

# Calculation of evaluation indicators
is = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

# Display of assessment results
print("\nLinear Regression Model Evaluation Results:")
print(f"Mean Absolute Error (MAE): {mae:.4f}")
print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.4f}")
print(f"R² Score: {r2:.4f}")
```

Step 3.Visualize the results

```
!pip install statsmodels
import statsmodels.api as sm

# Add constants to X_train to match statsmodels
X_train_sm = sm.add_constant(X_train)

# Building OLS models using statsmodels
model_sm = sm. OLS(y_train, X_train_sm).fit()

# Show detailed regression statistics table
print("\nLinear Regression Results Table (statsmodels):")
print(model_sm.summary())
```

8.2 Decision Tree Model

Step 1. Model Training

```
from sklearn.tree import DecisionTreeRegressor

# Initialize a Decision Tree model with a maximum depth
of 5 (adjustable)
model = DecisionTreeRegressor(max_depth=5,
random_state=42)
model.fit(X_train, y_train)

# Check the importance of input variables
Importance = model.feature_importances_
feature_names = X.columns

# Include results in a DataFrame for easy observation
importance_df = pd. DataFrame({
    'Feature': feature_names,
    'Importance': Importance
}).sort_values(by='Importance', ascending=False)

print("\nFeature Importances:")
print(importance_df)
```

Step 2. Model Evaluation

```
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score
import numpy as np

# Forecast on the test set
y_pred = model.predict(X_test)

# Calculation of model performance evaluation indicators
is = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

# Show Results
print("\n Decision Tree Model Evaluation Results:")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R² Score: {r2:.2f}")

# Compare actual and forecast values for the first 10 lines
comparison_df = pd. DataFrame({
    'Actual': y_test.values[:10],
    'Predicted': y_pred[:10]
})
print("\n Compare actual and forecast values (first 10
lines):")
print(comparison_df)
```

Step 3.Visualize the results

```
import matplotlib.pyplot as plt

# Draw a line chart to compare Actual vs Predicted
plt.figure(figsize=(10, 6))
plt.plot(range(len(comparison_df)),
comparison_df['Actual'], marker='o', color='black',
label='Actual')
plt.plot(range(len(comparison_df)),
comparison_df['Predicted'], marker='x', color='red',
label='Predicted')
```

```
plt.title('Actual vs Predicted ROI (Decision Tree)')
plt.xlabel('Sample Index')
plt.ylabel('ROI')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

8.3 Random Forest Model

Step 1. Model Training

```
from sklearn.ensemble import RandomForestRegressor

# Random Forest model initialization
model = RandomForestRegressor(n_estimators=100,
max_depth=7, random_state=42)
model.fit(X_train, y_train)

# Importance of Input Variables
Importance = model.feature_importances_
feature_names = X.columns

# Inserting into a DataFrame
importance_df = pd. DataFrame({
   'Feature': feature_names,
   'Importance': Importance
}).sort_values(by='Importance', ascending=False)

print("\nFeature Importances - Random Forest:")
print(importance_df)
```

Step 2. Model Evaluation

```
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score
import numpy as np

# Forecast on the test set
y_pred = model.predict(X_test)

# Calculation of evaluation indicators
is = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print("\n Random Forest Model Evaluation Results:")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R² Score: {r2:.2f}")

# Comparison of forecast and reality for the top 10 lines
comparison_df = pd. DataFrame({
   'Actual': y_test.values[:10],
   'Predicted': y_pred[:10]
})
print("\n Compare actual and forecast values (first 10
lines):")
print(comparison_df)
```

Step 3.Visualize the results

```
import matplotlib.pyplot as plt

# Draw a line chart comparing forecasted and actual values
plt.figure(figsize=(10, 6))
plt.plot(range(len(comparison_df)),
comparison_df['Actual'], marker='o', color='black',
label='Actual')
plt.plot(range(len(comparison_df)),
comparison_df['Predicted'], marker='x', color='green',
label='Predicted')
plt.title('Actual vs Predicted ROI (Random Forest)')
plt.xlabel('Sample Index')
plt.ylabel('ROI')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

V. Conclusion

This study presented a fundamental process for business data analysis using the Python programming language, covering key stages such as data preprocessing, exploratory data analysis, machine learning model development, and performance evaluation. By illustrating the process with real-world datasets and accessible Python code snippets, the paper offers both foundational knowledge in data analytics and practical guidance for students and learners in the fields of data science and digital business.

The application of three widely used regression models—Linear Regression, Decision Tree, and Random Forest—enables learners to understand how to select and assess suitable models within a business data context. Furthermore, the study highlights the critical role of preprocessing steps, including normalization, encoding, and feature transformation, which directly influence model performance and predictive accuracy. It is hoped that this document will serve as a valuable practical resource, helping students strengthen their data analysis competencies and enhance their ability to apply Python to real-world problems in business and management.

Acknowledgements

Financial support received from Thuongmai University is acknowledged.

Disclosure

Declaration of AI Assistance in Language Editing: The authors used ChatGPT to improve language clarity and correct minor errors in grammar and style. They reviewed all suggestions and are fully responsible for the content of this publication.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1]. Chen, B., & Kling, G. (2025). *Business analytics with Python: Essential skills for business students*. Kogan Page.

[2]. Gkikas, D. C., & Theodoridis, P. K. (2024). Predicting online shopping behavior: Using machine learning and Google Analytics to classify user engagement. *Applied Sciences, 14*(23), 11403. https://doi.org/10.3390/app142311403

[3]. Hodeghatta, U. R., & Nayak, U. (2023). *Practical business analytics using R and Python: Solve business problems using a data-driven approach* (2nd ed.). Apress.

[4]. Pes, B. (2021). Learning from high-dimensional and class-imbalanced datasets using random forests. *Information, 12*(8), 286. https://doi.org/10.3390/info12080286

[5]. Takale, S., Bhong, T., Dethe, U., & Gandhi, P. (2022). Sales prediction using linear regression. *Journal of Electronics, Computer Networking and Applied Mathematics, 2*(5), 62–71. https://doi.org/10.55529/jecnam.25.62.71

[6]. Zhou, Y., Ahmad, Z., Alsuhabi, H., Yusuf, M., Alkhairy, I., & Sharawy, A. M. (2021). Impact of YouTube advertising on sales with regression analysis and statistical modeling: Usefulness of online media in business. *Computational Intelligence and Neuroscience, 2021*, Article ID 9863155. https://doi.org/10.1155/2021/9863155