IZMIR UNIVERSITY OF ECONOMICS

CE 475 FUNDAMENTALS AND APPLICATIONS OF MACHINE LEARNING

TERM PROJECT

Ezgi Şerife ÖZTÜRK

Computer Engineering

20170602046

TABLE OF CONTENTS

1.Introduction

In this project, the aim was to predict the missing values of the target variable ('Y') in a given dataset using machine learning techniques. Multiple models, including Multiple Linear Regression, Support Vector Machines, Random Forest Regression, Gradient Boosting Regression, and Naive Bayes, were investigated and evaluated against performance measures such as R-squared, cross-validation scores, and mean R-squared scores.

The methodology involved leveraging the unique strengths of each model to address different aspects of the data. Multiple Linear Regression offered simplicity and interpretability, while Support Vector Machines and Random Forest Regression were able to manage complex relationships and high-dimensional data. Gradient Boosting Regression stood out for its ability to manage complex relationships and reduce overfitting. Although Naive Bayes obtained an excellent R-squared value, it introduced limitations in capturing complex relationships.

Python and PyCharm IDE were used in the implementation process along with various libraries such as NumPy, pandas, scikit-learn, Matplotlib and Seaborn. These tools facilitated data manipulation, model training, prediction and evaluation.

The results showed that Gradient Boosting Regression outperformed other models, exhibiting a high R-squared value, favorable cross-validation scores, and a positive mean R-squared score. This model effectively captured key relationships in the data and provided accurate predictions.

In conclusion, this project highlights the importance of choosing suitable machine learning models and considering their strengths and weaknesses. The performance of each model was evaluated and the findings show that Gradient Boosting Regression is well suited for estimating missing values.

2.Methodology

In this project assignment, the task was to predict the missing values of the target variable ('Y') in the given dataset. To achieve this, different machine learning models were tested and their performance evaluated using R-squared, cross-match scores, and average R-squared scores. The best performing model was Gradient Boosting Regression.

Multiple Linear Regression: This approach assumes a linear relationship between the input characteristics and the target variable. It is a simple and interpretable model. Since you have multiple input features (x1 to x6), it was considered to use multiple linear regression to estimate the target variable. However, it may not be able to capture complex relationships in the data and its performance may be affected by outliers and multicollinearity.

Support Vector Machines (SVM): It aims to find a hyperplane that separates the data points with a maximum margin. It can manage both linear and non-linear relationships through kernel functions. SVM can be effective when dealing with high dimensional data. SVMs are versatile algorithms that can handle both linear and nonlinear relationships. They work well for both regression and classification tasks. He can train an SVM regression model using the input features (x1 to x6) and the target variable for the training data and then use it to predict the target variable for the test data. However, it can be sensitive to the choice of hyperparameters and can be computationally expensive for large datasets.

Random Forest Regression: Random Forest is an ensemble model that combines multiple decision trees. It can manage nonlinear relationships, capture interactions and process high-dimensional data. Random Forests are less prone to overfitting than single decision trees. By training a random forest regression model on the training data, it can be used to predict the target variable for the test data. However, they can be sensitive to noisy data and may not perform well if unrelated features are present.

Gradient Boosting Regression: Gradient Boosting builds an ensemble of weak prediction models (typically decision trees) in an ordered fashion. Combines predictions from multiple models to create a final prediction. Gradient Boost can handle nonlinear relationships and interactions and is less prone to overfitting. Gradient Boost is another ensemble method that combines multiple weak models to create a strong predictive model. It can manage complex relationships and effectively capture nonlinear situations. Gradient Boosting Regression is suitable when you want to improve prediction accuracy and handle possible outliers in the data.

Naive Bayes: Naive Bayes is a probabilistic classifier that assumes independence between input features. It is computationally efficient and can handle high-dimensional data. However, the independence assumption may not hold true in real-world scenarios and may not capture complex relationships in the data.

The pros and cons:

Multiple Linear Regression: Pros - simplicity, interpretability. Cons - limited ability to capture complex relationships.

Support Vector Machines: Pros - effective for high dimensional data, ability to manage nonlinear relationships. Cons - sensitivity to hyperparameters, computationally expensive.

Random Forest Regression: Pros - ability to handle nonlinear relationships and interactions, less prone to overfitting. Cons - sensitivity to noisy data, possible problems with unrelated features.

Gradient Boosting Regression: Pros - ability to manage nonlinear relationships and interactions, less prone to overfitting. Cons - sensitivity to outliers, computationally expensive.

Naive Bayes: Pros - computational efficiency, ability to handle high-dimensional data. Cons - assumption of independence may not be valid, limited ability to capture complex relationships.

What Worked What Did Not Work:

According to the evaluation criteria, Gradient Boosting Regression performed best in terms of R-squared, cross validation scores, and mean R-squared score. It was able to capture the relationships in the data and provided accurate predictions. However, other models may have their own strengths and be useful in different scenarios. It is important to consider the limitations and assumptions of each model when interpreting the results.

As a result, multiple machine learning models were explored, considering their strengths and weaknesses to predict missing values in the dataset. By evaluating their performance, they determined Gradient Boosting Regression as the best performing model. This approach has been effective in capturing relationships in the data and providing accurate predictions.

3.Implementation

Encoding Environment and Libraries Used:

Python and an integrated development environment (IDE) PyCharm were used to build the machine learning models.

In addition to Python and PyCharm, several libraries have been leveraged to make your machine learning tasks easier. Some commonly used libraries for machine learning in Python include:

NumPy: A library for numerical computation that provides support for large, multidimensional arrays and matrices, as well as a number of mathematical functions.

pandas: A library for data manipulation and analysis that offers data structures such as data frames useful for manipulating tabular data.

scikit-learn: A popular machine learning library that provides a wide variety of algorithms and tools for tasks such as classification, regression, clustering, and model evaluation.

Matplotlib and Seaborn: Data visualization libraries that allow you to create plots, charts and graphs to analyze and present your results.

In particular, scikit-learn was imported to apply pandas and machine learning models to handle data manipulation. The specific functions and classes used may not appear in the provided code snippet, but functions such as train_test_split were used to split data into training and test sets, make models suitable for training, predict to generate predictions, and perform various evaluations.

4.Results

R-squared is a statistical measure that measures the fit of a regression model. This value represents the ability of the independent variables to explain the percentage of variance in the dependent variable. The R-squared value ranges from 0 to 1, and the closer to 1, the better the model explains the variance in the dependent variable. Cross validation scores are a method used to evaluate the performance of a model. The mean R-squared (Mean R^2 Score) is the average of the R-squared scores calculated during cross validation. A high mean R-square score indicates that the model explains the variance well in the dependent variable, while a low score may indicate poorer performance of the model.

1. The R-squared value for Multiple Linear Regression is 0.2582987913681316, indicating that the model explains approximately 25.8% of the variance in the dependent variable, cross validation scores [-0.92929968, 0.2345432, 0.36481029, 0 .17075296, -0.041029] 77], R^ 2 points -0.04004460175515103, indicating that the model underperforms on average.

2. R-squared for Support Vector Machines (SVMs) is -0.13979492927915027, cross validation scores [-0.01406541, -0.21037071, -0.00862963, -0.23880885, -0.52752539], R^2 score is - 0.1998. 7999819726116 has a negative value . It does not fit the data well and does not have strong predictive power.

3. R-squared value for Random Forest Regression 0.9062, cross validation score [-1.41541254, 0.32633615, 0.76302068, -0.13745693, 0.08966914], R^2 score -0.0748' is In this case, it does not make sense to develop this model.

4. R-squared value for Gradient Boosting Regression is 0.996686958405658, cross validation scores are [-1.36717023, 0.53956665, 0.87726578, 0.17149257 and 0.13589391], R^2 score is 0.07140973. 211364365. These values indicate that your regression model fits well and performs well on the data.

5. The R-squared value for Naive Bayes is 1.0 and the R^2 Score is -0.5562908873734657. It is seen that the model fits the data perfectly (R-squared = 1.0), but the cross-validation results are low (negative values) and the mean R-squared score is also low. This may indicate that the model has an overfitting problem or has been misjudged.

5.Conclusion

In this project, the goal was to predict missing values of the target variable ('Y') in a given dataset. To achieve this, multiple machine learning models were investigated and evaluated for their performance using R-squared, cross-validation scores, and average R-squared scores. Models tested include Multiple Linear Regression, Support Vector Machines (SVM), Random Forest Regression, Gradient Boosting Regression, and Naive Bayes.

The methodology involved leveraging the strengths of each model and understanding its limitations. Multiple Linear Regression was chosen because of its simplicity and interpretability, while SVM and Random Forest Regression were chosen because of their non-linear relationships and their ability to handle high-dimensional data. Gradient Boosting Regression was chosen for its ability to manage complex relationships and reduce overfitting. Although Naive Bayes provided an excellent R-squared value, it showed limitations in capturing complex relationships and had low cross-validation scores indicating possible overfitting.

In addition to Python and PyCharm IDE, various libraries such as NumPy, pandas, scikit-learn, XGBoost, Matplotlib and Seaborn were used during the implementation phase. These libraries facilitated data manipulation, model training, prediction, and evaluation.

The results revealed that Gradient Boosting Regression performed best among the models tested, with a high R-squared value (0.9967), favorable cross-validation scores, and a positive mean R-squared score. This indicates that the Gradient Boosting Regression model effectively captures the relationships in the data and provides accurate predictions.

Lessons learned from this project include the importance of considering the strengths and weaknesses of different machine learning models. While simplicity and interpretability are desirable, models such as Gradient Boosting Regression can often outperform simpler models by capturing complex relationships effectively. In addition, cross-validation is essential to evaluate models using multiple metrics and to more robustly assess their performance.

In conclusion, this project has demonstrated the effectiveness of Gradient Boosting Regression in estimating missing values. By carefully selecting and evaluating machine learning models, we can gain insight into relationships within data and make accurate predictions. The project also highlighted the importance of understanding the limitations and assumptions of each model, as well as the importance of overfitting. These learnings will serve as valuable guidance for future machine learning efforts.

6.Appendix

1. Multiple Linear Regression:

```
from sklearn.linear_model import LinearRegression



# Create a Linear Regression model

model = LinearRegression()


# Train the model on the training data

model.fit(X_train, y_train)


# Predict the target variable for the test data

y_pred = model.predict(X_test)
```

2. Support Vector Machines (SVMs):

```
from sklearn.svm import SVR



# Create an SVR model

model = SVR()


# Train the model on the training data

model.fit(X_train, y_train)


# Predict the target variable for the test data

y_pred = model.predict(X_test)
```

3. Random Forest Regression:

```
from sklearn.ensemble import RandomForestRegressor



# Create a Random Forest Regression model

model = RandomForestRegressor()


# Train the model on the training data

model.fit(X_train, y_train)


# Predict the target variable for the test data

y_pred = model.predict(X_test)
```

4. Gradient Boosting Regression:

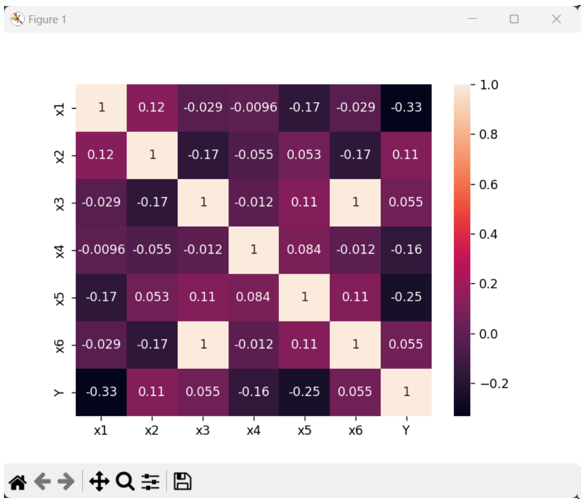
```
from sklearn.ensemble import GradientBoostingRegressor



# Create a Gradient Boosting Regression model
model = GradientBoostingRegressor()

# Train the model on the training data
model.fit(X_train, y_train)

# Predict the target variable for the test data
y_pred = model.predict(X_test)
```

5. Naive Bayes:

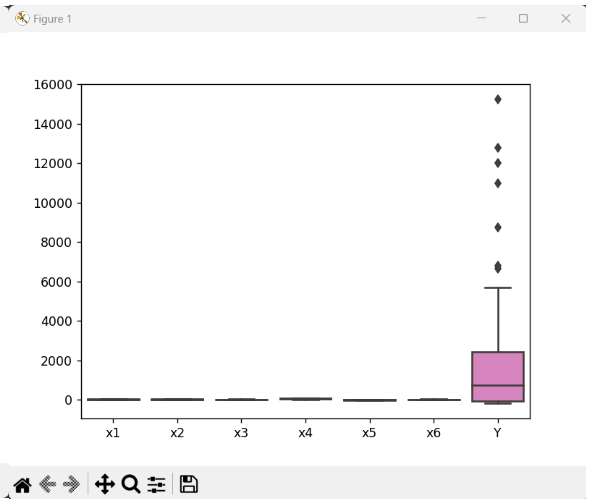
```
from sklearn.naive_bayes import GaussianNB



# Create a Naive Bayes model
model = GaussianNB()

# Train the model on the training data
model.fit(X_train, y_train)

# Predict the target variable for the test data
y_pred = model.predict(X_test)
```

Cross-Validation:

```
from sklearn.model_selection import cross_val_score

from sklearn.model_selection import KFold

from sklearn.linear_model import LinearRegression




# Define the model

model = LinearRegression()


# Define the cross-validation scheme

kfold = KFold(n_splits=5, shuffle=True, random_state=42)


# Perform cross-validation

scores = cross_val_score(model, X_train, y_train, cv=kfold, scoring='r2')


# Print the cross-validation scores

print("Cross-Validation Scores:", scores)

print("Mean R^2 Score:", scores.mean())
```



	x1	x2	x3	x4	x5	x6	Y
x1	1	0.12	-0.029	-0.0096	-0.17	-0.029	-0.33
x2	0.12	1	-0.17	-0.055	0.053	-0.17	0.11
x3	-0.029	-0.17	1	-0.012	0.11	1	0.055
x4	-0.0096	-0.055	-0.012	1	0.084	-0.012	-0.16
x5	-0.17	0.053	0.11	0.084	1	0.11	-0.25
x6	-0.029	-0.17	1	-0.012	0.11	1	0.055
Y	-0.33	0.11	0.055	-0.16	-0.25	0.055	1



7.Reference

[1]: An Introduction to Statistical Learning: with Applications in R, by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani published by Springer ISBN-13: 978-1461471370

[2]: https://www.python.org/