BIRZEIT UNIVERSITY

Faculty of Engineering and Technology

Department of Electrical and Computer Engineering

ENCS5141—Intelligent Systems Laboratory

Assignment #1—Comparing the Time Complexity of Binary Search and Linear Search algorithms

Prepared by: Tala Flaifel 1201107

Instructor: Mohammad Jubran

Teacher Assistant: Hanan Awawdeh

Date: March 31, 2024

Abstract

This case study focuses on the critical rule of data preprocessing preparing the Titanic dataset for machine learning analysis, the study will thoroughly explore preprocessing steps to optimise and improve the dataset performance, by handling the missing values, outliers, scaling the data, ensuring feature relevance.

Table of content

Introduction

Data processing is the first essential step in transforming and changing raw data to be usable and understandable by the machine. It's the idea of organising data into a clear and concise summary, the extraction of maximum value and insight from the dataset, like sifting through a pile of unrecognisable notes to find and point the key points.

The importance of data preprocessing cannot be overemphasised; it helps create a solid foundation for which machine learning models would be trained on. By fully studying the data and addressing the quality issues, and properly dealing with the issues to transform the raw data into a fully understandable, and manageable format for machine learning analysis.

Figure 1.1 Data Processing.

The study's ultimate goal is to illuminate the factors that influenced survival on the Titanic by accurately preparing the dataset for machine learning analysis. Through careful data cleaning and feature engineering

Throughout the case study, a systematic approach will be taken, starting with initial data exploration to gain a deep understanding of the dataset's structure, statistics and characteristics. Will proceed to look into data quality issues, outliers, feature relevance, as well as encoding categorical features.

Furthermore, scaling, normalisation, and dimensionality reduction techniques will be employed to fully process the dataset. Finally, the study will end with validating the preprocessing pipeline by training and evaluating a machine learning model on the preprocessed data, comparing its performance against a model trained on the raw dataset.

The objective of this experiment is to compare the outcome of model training on processed and unprocessed data, to verify the expected results of significantly better results using the processed data.

Procedure and Discussion

The study starts with reading and printing the dataset to gain an insight into the data's features, then will be looking into more information about the dataset; get the data structure and statistical descriptions of the numerical values. Listing 2.1 shows how the information is being extracted in python.

```
print(df.shape)
df.describe()
df.info()
```

Listing 2.1 statistical data extraction.

The result shows a foundational understanding of the dataset's characteristics. The data consists of 891 rows and 15 columns. It shows that female survival is higher than men - class 1 and 2 is high in comparison with class 1 with a very low survival rate.

Now will be looking into the missing values and handling them, Listing 2.2 shows the search for missing values.

```
df.isnull().sum()
```

Listing 2.2 missing values code snippet.

The results show a 77% of missing values in the deck feature, and 19% of missing values in the age feature, as well as a .2% from embarked and emabarked_town.

Will be dropping the deck feature to deal with the huge proportion of the deck feature missing, and will use imputation with median to deal with the age feature to help preserve the distribution of the feature and be less sensitive to outliers. For both embarked and embark_town features both will be dropped considering both missing values are on the same row, and the low impact they have. Listing 2.3 shows the code snippet that dealt with them.

```
df.drop('feature', axis=1, inplace=True)
# Impute missing 'age' values with median
median_age = df['age'].median()
df['age'].fillna(median_age, inplace=True)
```

Listing 2.3 handling missing values.

Now in order to identify potential outliers and gain valuable insights into the distribution of numerical variables in a dataset will use the interquartile range (IQR) method, as well as data visualisation.

```
columns = ['survived', 'pclass', 'age', 'sibsp', 'fare']

# Loop through each column and compute percentiles and IQR
for column in columns:
    # Compute percentiles
    percentile_25 = df[column].quantile(0.25)
    percentile_75 = df[column].quantile(0.75)

    print("Column:", column)
    print("25th Percentile:", percentile_25)
    print("75th Percentile:", percentile_75)

    # Compute interquartile range (IQR)
    iqr = percentile_75 - percentile_25
    print("Interquartile Range (IQR):", iqr)
    print()
```

Listing 2.4 IQR.

```
sns.boxplot(x = "age", data = df)
```

Listing 2.5 Visualisation for age feature to spot outliers.

After handling the missing data, an increase in outliers was found. To further investigate the outliers will be employing the IQR rule specifically for the age feature.

```
NumRecordsBefore=df.shape[0]
DroppedRecords=df[(df['age'] < LowerBound_age) | (df['age'] >
UpperBound_age)].shape[0]
print(f"Number of outliers based on the Interquartile Range and Boxplots is
{DroppedRecords} ({100*DroppedRecords/NumRecordsBefore}%)")
```

Listing 2.6 IQR for age feature.

```
Q1 = 22.0  # 25th percentile
Q3 = 35.0  # 75th percentile
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
df["age"] = np.clip(df["age"], lower_bound, upper_bound)
```

Listing 2.7 Handling outliers for the age feature.

Listing 2.7 shows the outliers handling for the age feature, by employing the clipping technique by constraining the outliers to fall within the defined range. This resulted in zero outliers in the feature.

In order to have all the data in a format understandable by the machine, encoding all the categorical features is required. And will be employing one hot encoding for its advantage in preserving the information without any hierarchy or ordinality. As shown in listing 2.8.

```
df_encoded = pd.concat([df, pd.get_dummies(categorical_features,
dtype=int)], axis=1)
```

Listing 2.9 One hot encoding on categorical features.

Will be studying the feature relevance inorder to retain the most relevant features for an improved model performance and robustness using f-regression for information gain. Shown in listing 2.9.

```
selector = SelectKBest(score_func=f_regression, k=10)
selector.fit(df_encoded, df_encoded[target_col])
feature_scores = selector.scores_
```

Listing 2.9 Find feature relevance.

The result showed a high relevance for almost 15 of the features, the rest were dropped for a more efficient data processing.

To ensure that all features contribute equally to the model's learning process scaling the data is a must. Shown in Listing 2.10.

```
scaler = MinMaxScaler()
features_to_scale = ['age', 'fare','parch','pclass']
scaler.fit(X_train[features_to_scale])
X_train[features_to_scale] = scaler.transform(X_train[features_to_scale])
X_test[features_to_scale] = scaler.transform(X_test[features_to_scale])
```

Listing 2.10 MinMax scaling

The scaling results in data regularisation and normalisation with MinMax scaler.

For a simpler, faster, and an improved model performance, a lower chance of overfitting, reducing dimensionality is required. Principal Component Analysis (PCA) is the technique used to ensure preserving as much variance as possible. As in listing 2.11.

```
pca = PCA(n_components=10)
pca.fit(X_train)
Array_PCA = pca.transform(X_train)
```

```
X_test_pca = pca.transform(X_test)
```

Listing 2.10 PCA.

Finally to validate the process of data preprocessing, a comparison using model performance on the processed data vs the raw data will be held, by training both models using Support Vector Machine (SVM) training model. In listing 2.11.

SVM is sensitive to feature scaling and can be affected by outliers, so preprocessing steps like scaling and outlier removal would have a pronounced impact on its performance.

```
# Model with original features
clf_original = SVC(random_state=42)
clf_original.fit(or_X_train, or_y_train)
y_pred_original = clf_original.predict(or_X_test)
accuracy_original = accuracy_score(or_y_test, y_pred_original)

# Model with PCA components
clf_pca = SVC(random_state=42)
clf_pca.fit(Array_PCA, y_train)  # Note: Use PCA-transformed training data
y_pred_pca = clf_pca.predict(X_test_pca)  # Note: Use PCA-transformed test data
accuracy_pca = accuracy_score(y_test, y_pred_pca)
```

Listing 2.11 Model training unprocessed and raw data.

```
Accuracy with original features (SVM): 0.7191011235955056
Accuracy with PCA components (SVM): 1.0
```

Figure 2 Comparison result.

The results demonstrate that preprocessing the data significantly improved the performance of the SVM classifier compared to the model performance on the raw data, which is proof enough that data preprocessing is a very important step in machine learning analysis due to the huge impact it has on the training outcome.

The results of the raw data 71% is considerably good for a model training, but when in comparison with the perfect results of 100% that the processed data resulted in, it is a no brainer to understand and appreciate the importance of data preprocessing in machine learning analysis.

Conclusion

The case study demonstrated the importance and power of data preprocessing for machine learning analysis, by handling the quality issues, the missing values, transforming the categorical data, reducing dimensionality, data scaling, and feature relevance selection all resulted in an impressive results in model training with an accuracy of 100%. The comparison between the raw data and preprocessed data performance showed the huge positive impact data preprocessing made. By carefully cleaning and engineering the Titanic dataset, a strong foundation was laid for building robust models that illuminates the survival rate factors.