Data Collection and Preprocessing Phase

Date	8 July 2024
Team ID	SWTID1720092248
Project Title	Revolutionizing Liver Care: Predicting Liver Cirrhosis Using Advanced Machine Learning Techniques
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

Section	**Description**
Data Overview	The data overview provides basic statistics, dimensions, and structure of the dataset. It includes the number of records, number of features, and data types of each feature
Univariate Analysis	Univariate analysis involves exploring individual variables to understand their distribution, central tendency (mean, median, mode), and dispersion (variance, standard deviation). Visualizations such as histograms and box plots are used to illustrate these statistics
Bivariate Analysis	Bivariate analysis examines the relationship between two variables. This includes calculating correlation coefficients and creating scatter plots to visualize potential linear or non-linear relationships between pairs of variables
Multivariate Analysis	Multivariate analysis explores patterns and relationships involving multiple variables simultaneously. Techniques such as principal component analysis (PCA) and multiple regression analysis are employed to understand the interactions between variables
Outliers and Anomalies	Identifying and treating outliers is crucial to ensure accurate analysis. This section involves detecting outliers using

<table>
<tr><td></td><td>statistical methods (e.g., IQR, Z-scores) and deciding on appropriate treatments (e.g., removal, transformation)</td></tr>
<tr><td colspan="2">Data Preprocessing Code Screenshots</td></tr>
<tr><td>Loading Data</td><td><pre>#Reading csv file
df = pd.read_csv('HealthCareData.csv')
df.head()</pre></td></tr>
<tr><td>Handling Missing Data</td><td><pre># Drop columns with more than a certain percentage of missing values (e.g., 50%)
threshold = len(df) * 0.5
df = df.dropna(axis=1, thresh=threshold)

# Fill numerical columns with mean
numerical_cols = df.select_dtypes(include=[np.number]).columns
df[numerical_cols] = df[numerical_cols].fillna(df[numerical_cols].mean())

# Fill categorical columns with mode
categorical_cols = df.select_dtypes(include=[object]).columns
df[categorical_cols] = df[categorical_cols].apply(lambda x: x.fillna(x.mode()[0]))

Check remaining nulls
print(df.isnull().sum())</pre></td></tr>
<tr><td>Data Transformation</td><td><pre>from sklearn.preprocessing import LabelEncoder</pre>
<pre>le=LabelEncoder()
for col in categorical_cols:
    df[col] = le.fit_transform(df[col])

df.head()</pre></td></tr>
<tr><td>Feature Engineering</td><td>Perform Chi-Square Test for Categorical Variables
<pre># Function to perform Chi-Square Test
def chi_square_test(feature):
    contingency_table = pd.crosstab(data[feature], data[target])
    chi2, p, dof, ex = chi2_contingency(contingency_table)
    return p

# Apply Chi-Square Test to categorical features
chi_square_results = {feature: chi_square_test(feature) for feature in categorical_features}

Display Chi-Square Test results
chi_square_results</pre>
Perform ANOVA for Continuous Variables
<pre># Function to perform ANOVA
def anova_test(feature):
    groups = [data[data[target] == level][feature] for level in data[target].unique()]
    f_val, p_val = f_oneway(*groups)
    return p_val

# Apply ANOVA to continuous features
anova_results = {feature: anova_test(feature) for feature in continuous_features}

Display ANOVA results
anova_results</pre></td></tr>
</table>

<table>
<tr><td></td><td>

Combine Results

```
# Combine results
all_results = {**chi_square_results, **anova_results}

# Display all results
all_results
```

</td></tr>
<tr><td>Save Processed Data</td><td>

Filter Significant Features

```
significance_level = 0.05 #(alpha-value)assumption
significant_features = [feature for feature, p_value in all_results.items() if p_value < significance_level]

# Display significant features
significant_features
```

```
df = df[significant_features].copy()
df.head()
```

</td></tr>
</table>