Feature Engineering and Selection

Overview

Feature engineering and selection are critical steps in the data preprocessing pipeline for machine learning. They involve creating new features or modifying existing ones to improve model performance and selecting the most relevant features to enhance predictive accuracy and reduce overfitting.

Why Use Feature Engineering and Selection?

- **Improved Model Performance**: Enhancing and selecting features can lead to better predictive accuracy.
- **Dimensionality Reduction**: Reducing the number of features simplifies models, making them faster and more interpretable.
- **Noise Reduction**: Eliminating irrelevant features decreases noise, leading to more robust models.

Prerequisites

- **Python**: Ensure Python is installed on your system.
- **Pandas**: For data manipulation and analysis.

```
pip install pandas
```

- **NumPy**: For numerical operations.

```
pip install numpy
```

- **Scikit-learn**: For machine learning tools and algorithms.

```
pip install scikit-learn
```

- **Seaborn and Matplotlib**: For data visualization.

```
pip install seaborn matplotlib
```

Files Included

- `feature_engineering_selection.py` : Contains the code for performing feature engineering and selection on the Titanic dataset.
- `Titanic-Dataset.csv` : The dataset used for analysis.
- `README.md` : This documentation file.

Code Description

The provided code demonstrates various feature engineering and selection techniques applied to the Titanic dataset.

1. Importing Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxSca
from sklearn.feature_selection import VarianceThreshold, SelectKBest, chi
from sklearn.linear_model import LogisticRegression, Lasso
from sklearn.ensemble import RandomForestClassifier
from sklearn.decomposition import PCA
```

- **Pandas and NumPy**: For data manipulation and numerical operations.
- **Seaborn and Matplotlib**: For data visualization.
- **Scikit-learn Modules**: For preprocessing, feature selection, and modeling.

2. Loading and Inspecting the Dataset

```
# Load the Titanic dataset
file_path = "Titanic-Dataset.csv"
df = pd.read_csv(file_path)

# Display first 5 rows
df.head()

# Check missing values
print(df.isnull().sum())
```

- **Loading Data**: Reads the Titanic dataset into a DataFrame.
- **Inspecting Data**: Displays the first five rows and checks for missing values.

3. Handling Missing Values

```
# Fill missing numerical values with median
num_imputer = SimpleImputer(strategy='median')
df[['Age', 'Fare']] = num_imputer.fit_transform(df[['Age', 'Fare']])

# Fill missing categorical values with most frequent value
cat_imputer = SimpleImputer(strategy='most_frequent')
df[['Embarked']] = cat_imputer.fit_transform(df[['Embarked']])

# Drop Cabin column due to too many missing values
df.drop(columns=['Cabin'], inplace=True)

# Verify missing values are handled
print(df.isnull().sum())
```

- **Numerical Imputation**: Fills missing values in 'Age' and 'Fare' with the median.
- **Categorical Imputation**: Fills missing 'Embarked' values with the most frequent category.
- **Dropping Columns**: Removes the 'Cabin' column due to excessive missing values.

4. Encoding Categorical Variables

```
# Label Encoding for binary categorical feature
label_encoder = LabelEncoder()
df['Sex'] = label_encoder.fit_transform(df['Sex'])
```

```
# One-Hot Encoding for multi-category features
df = pd.get_dummies(df, columns=['Embarked'], drop_first=True)
```

- **Label Encoding**: Converts the binary 'Sex' feature into numerical form.
- **One-Hot Encoding**: Transforms the 'Embarked' feature into dummy variables.

5. Dropping Unnecessary Columns

```
# Drop unnecessary columns
df.drop(columns=['Name', 'Ticket'], inplace=True)
```

- **Dropping Columns**: Removes 'Name' and 'Ticket' as they are not useful for modeling.

6. Feature Scaling

```
# Standardization
scaler = StandardScaler()
df_scaled = df.copy()
df_scaled[['Age', 'Fare']] = scaler.fit_transform(df_scaled[['Age', 'Fare

# Display transformed dataset
df_scaled.head()
```

- **Standardization**: Scales 'Age' and 'Fare' to have a mean of 0 and a standard deviation of 1.

7. Feature Selection

a. Variance Threshold

```
# Variance Threshold to remove low variance features
selector = VarianceThreshold(threshold=0.01)
df_selected = df_scaled.copy()
df_selected = pd.DataFrame(selector.fit_transform(df_selected), columns=c

# Display selected features
df_selected.head()
```

- **Low Variance Feature Removal**: Eliminates features with variance below 0.01.

b. Chi-Square Test

```
# Define X and y
X = df_selected.drop(columns=['Survived'])  # Features
y = df_selected['Survived']  # Target variable

# Apply MinMaxScaler to ensure non-negative values
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Apply Chi-Square test
chi_selector = SelectKBest(score_func=chi2, k=5)
X_new = chi_selector.fit_transform(X_scaled, y)

# Get selected feature names
selected_features = X.columns[chi_selector.get_support()]
print("Selected Features:", selected_features)
```

- **Chi-Square Test**: Selects the top 5 features that have the strongest

---

Expected Output

Upon executing the provided code, the following outcomes are anticipated:

1. **Data Preprocessing**:
    - **Missing Values**: The 'Age' and 'Fare' columns will have missing values filled with their respective median values. The 'Embarked' column will have missing entries filled with the most frequent category.
    - **Column Removal**: The 'Cabin' column will be dropped due to a high percentage of missing values.
2. **Feature Encoding**:
    - **'Sex' Column**: Converted into numerical format using label encoding, where 'male' and 'female' are represented as 1 and 0, respectively.
    - **'Embarked' Column**: Transformed into dummy variables (one-hot encoding), resulting in new columns such as 'Embarked_Q' and 'Embarked_S'.
3. **Feature Scaling**:
    - **'Age' and 'Fare' Columns**: Standardized to have a mean of 0 and a standard deviation of 1.

4. **Feature Selection**:
    - **Variance Threshold**: Features with variance below 0.01 are removed.
    - **Chi-Square Test**: Top 5 features selected based on their chi-square statistics.
    - **Recursive Feature Elimination (RFE)**: Top 5 features identified using RFE with Logistic Regression.
    - **Lasso Regression**: Features with non-zero coefficients selected, indicating their importance.
    - **Random Forest Importance**: Features ranked based on their importance scores from a Random Forest classifier.
5. **Dimensionality Reduction**:
    - **PCA**: The dataset is reduced to two principal components, which are then visualized in a scatter plot, differentiating between survivors and non-survivors.

Use Cases

The techniques demonstrated in this code are applicable in various scenarios:

1. **Predictive Modeling**: Enhancing model performance by selecting and engineering features that have the most predictive power.
2. **Data Cleaning**: Handling missing values and removing irrelevant or redundant features to improve data quality.
3. **Dimensionality Reduction**: Reducing the number of features to simplify models, decrease computational cost, and mitigate overfitting.
4. **Exploratory Data Analysis (EDA)**: Understanding data distributions and relationships between variables to inform feature engineering decisions.
5. **Model Interpretability**: Identifying and selecting key features that contribute most to model predictions, aiding in the interpretability of the model.

Future Enhancements

To further improve the feature engineering and selection process, consider the following enhancements:

1. **Automated Feature Engineering**: Implement tools and frameworks that automatically generate and select meaningful features, reducing manual effort and potentially uncovering complex

feature interactions.

2. **Advanced Encoding Techniques**: Explore encoding methods like target encoding or embedding techniques for categorical variables, especially when dealing with high-cardinality features.
3. **Feature Interaction Analysis**: Investigate interactions between features to create new composite features that capture underlying patterns not apparent from individual features alone.
4. **Regularization Methods**: Apply regularization techniques such as Ridge or Elastic Net to prevent overfitting and enhance the generalization of the model.
5. **Cross-Validation Strategies**: Utilize robust cross-validation methods to assess the stability and reliability of selected features across different subsets of the data.

References

- Feature Engineering Explained | Built In
- Innovative Ways to Enhance ML Models with Feature Engineering
- How Does Feature Engineering Improve ML Algorithms? - H2O.ai
- A Short Guide for Feature Engineering and Feature Selection - GitHub
- Feature Engineering and Selection: A Practical Approach for Predictive Models

---