Documentations

Prosenjit Mondol

August 8, 2025

1 Data Preprocessing

- **Sampling**
  Sampling techniques are used to select a representative subset of data from a large population to reduce the computational complexity and improve the efficiency of the analysis.

- **Transformation**
  Transformation techniques involve manipulating raw data to create a single input, such as scaling, normalization, or encoding categorical data.

- **Denoising**
  Denoising techniques remove unwanted noise from the data that can lead to inaccurate results.

- **Imputation**
  Imputation techniques are used to fill in missing values in the data using statistical methods.

- **Feature extraction**
  Feature extraction techniques help to identify and extract relevant features from the data that are significant in a particular context.

- **Normalization**
  Normalization techniques are used to organize data for more efficient access and processing.

2 Handle Categorical Data

Categorical data is a type of data that represents qualitative or nominal characteristics, such as gender, occupation, Categorical data cannot be measured or compared using mathematical operations like addition or subtraction.

2.1 Different Encoding Methods for Categorical Data

- **One-Hot Encoding**
  One-Hot Encoding creates a new binary column for each category.

  Listing 1: Logistic Regression Example

```
X = pd.get_dummies(X)
print(X)
```

- **Label Encoding**
  Label Encoding assigns a numerical value to each category.

```
from sklearn.preprocessing import LabelEncoder

lencoders = {}

for col in data[features].columns:
    lencoders[col] = LabelEncoder()
    data[col] = lencoders[col].fit_transform(data[col])

data[features].nunique()
```

- **Binary Encoding**
  Binary Encoding creates new columns representing each category.

2.2 Looking at null or missing values

- **Mean Imputation**
  Mean imputation is a simple and widely used method for filling in missing values.
- **Mode Imputation**
  Mode imputation is a method for filling in missing values that is similar to mean imputation, but instead of using the mean, it uses the mode of the available values in a column.
- **K-Nearest Neighbor (KNN) Imputation**
  KNN imputation is a method for filling in missing values that is based on the distance between data poionts.

Listing 2: Logistic Regression Example

```
# Multiple Imputation by Chained Equations
from sklearn.experimental import
    enable_iterative_imputer
from sklearn.impute import IterativeImputer

#mputed_data = df[numerical_columns].copy(deep=
    True)
mice_imputer = IterativeImputer()
data[numerical_columns] = mice_imputer.
    fit_transform(data[numerical_columns])
```

3 Checking imblanced in target variable

- **Handling imbalanced data using oversampling**
  oversampling is a method for handling imbalanced data by increasing the size of the minority class.

Listing 3: Logistic Regression Example

```python
from sklearn.utils import resample

no = normalized_data[normalized_data.RainTomorrow ==
    0]
yes = normalized_data[normalized_data.RainTomorrow ==
     1]
yes_oversampled = resample(yes, replace=True,
    n_samples=len(no), random_state=123)
oversampled_data = pd.concat([no, yes_oversampled])

fig = plt.figure(figsize = (8,5))
sns.countplot(x='RainTomorrow', data =
    oversampled_data, palette = "Set1").set(title='
    RainTomorrow Indicator No(0) and Yes(1) after
    Oversampling (Balanced Dataset)')
```

- **How multicollinearity affects decision trees**
  Multicollinearity affects decision trees by reducing the importance and accuracy of the input features.

Listing 4: Logistic Regression Example

```python
    #the heat map of the correlation
    plt.figure(figsize=(16,10))
    sns.heatmap(X.corr(), annot=True, cmap='RdYlGn')
```

4 Outliner Detection

- **Boxplot Method**
  One of the simplest and most popular methods for detecting outliers is the box-plot.

Listing 5: Logistic Regression Example

```python
        plt.figure(figsize=(50,25))
        sns.boxplot(data=scaled_data[
            numerical_features])
```

- **Z-Score Method**
  The Z-Score method is a simple and widely used method for detecting outliers.

Listing 6: Logistic Regression Example

```
from scipy import stats
import numpy as np

# Calculate Z-scores for each value in the numerical
    features
z_scores = np.abs(stats.zscore(scaled_data[
    numerical_features]))

# Identify outliers (e.g., Z-score > 3)
outliers = (z_scores > 3)

# Print rows with outliers
print(scaled_data[outliers.any(axis=1)])
```

- **Transformation**
  Transformation involves transforming the data to a different scale to reduce the impact of the outliers.

Listing 7: Logistic Regression Example

```
        from sklearn.preprocessing import
            PowerTransformer
        # Apply Power Transformation to the numerical
             features
        power_transformer = PowerTransformer()
        scaled_data[numerical_features] =
            power_transformer.fit_transform(
            scaled_data[numerical_features])
```

5 Let's see how it fared in prediction using Logistic Regression

Listing 8: Logistic Regression Example

```
# Train a logistic regression model on the training set
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression

# Instantiate the model
logreg = LogisticRegression(solver='liblinear',
    random_state=0)
```

```python
# Fit the model
logreg.fit(X_train, y_train)

# Predict on the test set
y_pred_test = logreg.predict(X_test)

print('Model accuracy score: {0:0.4f}'.format(
    accuracy_score(y_test, y_pred_test)))
```