Comprehensive Data Science Documentation

Prosenjit Mondol
Data Scientist
prosenjit1156@email.com

September 7, 2025

Contents

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.

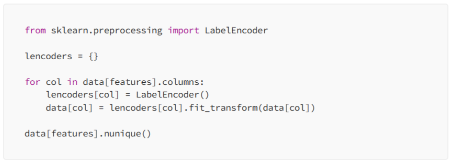
2.1 Different Encoding Methods for Categorical Data

- **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

```
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

```
    #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

```
            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

```python
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

```python
        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

```python
# 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)

# 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)))
```

6 Algorithms

6.1 Simple Linear Regression

The output is shown in the best fit line.

$$\begin{aligned} y &= mx + C \\ h_0(x) &= \theta_0 + \theta_1 x \\ h_0(x) &= \hat{y} \quad \text{(predicted value)} \\ error &= y - \hat{y} \end{aligned}$$

Here, θ_0 is the intercept, θ_1 is the slope or cofficient.
if $x = 0$, then $h_0(x) = \theta_0$ (intercept).

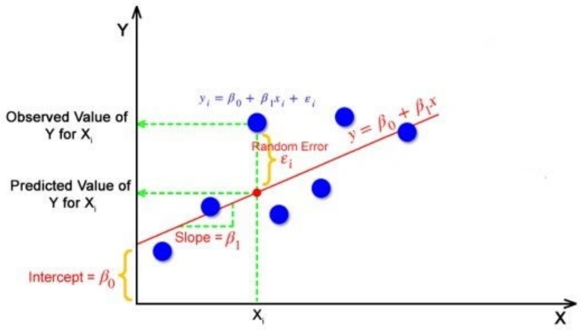

Y
X
Observed Value of Y for X_i
Predicted Value of Y for X_i
$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$
Random Error ε_i
$y = \beta_0 + \beta_1 x$
Slope = β_1
Intercept = β_0
X_i

Figure 1: Simple Linear Regression: Best Fit Line, Intercept, Slope, and Error

6.2 Convergence Algorithm *(Optimize the changes of theta_1 values)*

Repeat until convergence:

$$\theta_j = \theta_j - \alpha \cdot \frac{\partial J(\theta_j)}{\partial \theta_j}$$

7 Handle missing values

why it not

8 Models

8.1 Convolutional Neural Networks (CNNs)

Definition:
Convolutional Neural Networks (CNNs) are deep learning models designed for image analysis, using convolutional layers to automatically extract hierarchical spatial features, pooling layers to reduce dimensionality, and fully connected layers for classification or prediction.

8.1.1 Basic CNN

Definition:
A Basic CNN consists of convolutional layers, pooling layers, and fully connected layers, used mainly for simple image classification and recognition tasks.

8.1.2 ResNet50

Definition:
ResNet50 is a 50-layer Residual Network that uses skip connections to overcome vanishing gradients, enabling efficient training of deep networks for image classification and detection.

8.1.3 ResNet-101

Definition:
ResNet-101 is a deeper variant of ResNet with 101 layers. It employs residual connections to maintain gradient flow, allowing extraction of complex features for advanced classification and medical imaging tasks.

8.1.4 U-Net Model

Definition:
U-Net is a CNN architecture designed for biomedical image segmentation. Its encoder-decoder structure with skip connections enables precise, pixel-level segmentation of medical images such as MRI scans.

8.1.5 Faster R-CNN

Definition:
Faster R-CNN is an object detection framework that integrates Region Proposal Networks (RPN) with CNNs, enabling accurate localization and classification of objects within images.

8.1.6 CNN + LSTM

Definition:
CNN + LSTM combines convolutional networks for spatial feature extraction with Long Short-Term Memory (LSTM) networks for temporal sequence modeling, useful for multi-slice MRI analysis.

8.1.7 GAN (Generative Adversarial Network)

Definition:
GANs consist of a generator and discriminator network trained adversarially to produce realistic synthetic images, widely used for MRI data augmentation and balancing datasets.

8.1.8 Transformer-based CNNs

Definition:
Transformer-based CNNs integrate self-attention mechanisms with convolutional layers, enabling better global context understanding and higher accuracy in segmentation and classification tasks.

8.2 Visual Geometry Group (VGG)

Definition:
VGG is a deep CNN architecture characterized by small 3×3 convolutional filters stacked in depth. VGG models are widely used for image recognition due to their simplicity and effectiveness.

8.2.1 VGG16

Definition:
VGG16 is a variant with 16 layers (13 convolutional and 3 fully connected). It is simple yet effective for image classification and feature extraction.

8.2.2 VGG19

Definition:
VGG19 is a deeper variant with 19 layers (16 convolutional and 3 fully connected). Its increased depth enables capturing more complex features while maintaining architectural simplicity.

8.3 Ensemble Model

Definition:
Ensemble models combine multiple base models to improve prediction accuracy and robustness compared to individual models.

8.4 SOTA Model

Definition:
A State-of-the-Art (SOTA) model represents the best-performing deep learning architecture for a specific task at a given time, based on benchmarks or research outcomes.

8.5 Otsu Method

Definition:
Otsu's method is an automatic image thresholding technique that separates foreground and background by minimizing intra-class variance of pixel intensity.

8.6 Convolutional Neural Networks (CNNs)

Definition:
The study employs advanced Convolutional Neural Networks (CNNs) to address the challenge of accurately detecting and localizing spinal cord injuries (SCI) in MRI images. Two key CNN architectures were utilized: VGG-16 and ResNet50. These models were integrated into the Faster R-CNN framework, a robust object detection system that combines region proposal networks (RPN) with CNNs for feature extraction and object classification.

8.6.1 ResNet50

Definition:
ResNet (Residual Networks) is a more advanced architecture designed to address the vanishing gradient problem in deep networks. ResNet50 consists of 50 layers, incorporating residual connections, which allow gradients to bypass certain layers during backpropagation, effectively preventing the gradient from vanishing or exploding as the network depth increases.

8.6.2 ResNet-101

Definition:
ResNet-101 is a deeper variant of the Residual Network (ResNet) architecture consisting of 101 layers. Like other ResNet models, it incorporates residual connections, which allow gradients to bypass certain layers during backpropagation. This design effectively prevents the vanishing or exploding gradient problem, enabling the training of very deep networks. With its greater depth compared to ResNet-50, ResNet-101 is capable of capturing more complex and abstract features, making it highly effective for advanced image classification, object detection, and medical image analysis tasks.

8.6.3 U-Net Model

Definition:
U-Net is a convolutional neural network (CNN) architecture, initially designed for biomedical image segmentation, but has since been adopted for a wide range of tasks, including general image segmentation and even language modeling.

8.7 Visual Geometry Group (VGG)

Definition:
VGG (Visual Geometry Group) is a deep convolutional neural network architecture known for its effectiveness in image recognition tasks. It's characterized by its simple yet deep structure, utilizing small 3x3 convolutional filters repeatedly. VGG models, particularly VGG16 and VGG19, are widely used and have achieved notable performance on datasets like ImageNet.

8.7.1 VGG16

Definition:
VGG16 is a specific variant of the VGG architecture with 16 layers (13 convolutional layers and 3 fully connected layers). It is known for its depth and simplicity, making it effective for image classification tasks.

8.7.2 VGG19

Definition:
VGG19 is a specific variant of the VGG architecture with 19 layers (16 convolutional layers and 3 fully connected layers). It is known for its greater depth compared to VGG16, which allows it to capture more complex features in images. Like VGG16, it uses small 3×3 convolution filters and a simple, uniform architecture, making it highly effective for image classification and feature extraction tasks.

8.8 Ensemble Model

Definition:
An ensemble model in machine learning combines the predictions of multiple individual models (base estimators) to produce a more accurate and robust prediction than any single model alone.

8.9 SOTA Model

Definition:
In deep learning, SOTA model means State-of-the-Art model — basically, the best-performing architecture or method for a given task at a given time, according to benchmarks or competitions.

8.10 Utsu Method

Definition:
Otsu's method is a technique used in computer vision and image processing for automatic image thresholding.

9 Plotting