19_Pipeline

September 3, 2024

1 Pipeline in Machine Learning

In machine learning, a pipeline is a sequence of data processing steps that are chained together to automate and streamline the machine learning workflow. A pipeline allows you to combine multiple data preprocessing and model training steps into a single object, making it easier to organize and manage your machine learning code.

Here are the key components of a pipeline:

`Data Preprocessing Steps:` Pipelines typically start with data preprocessing steps, such as feature scaling, feature encoding, handling missing values, or dimensionality reduction. These steps ensure that the data is in the appropriate format and quality for model training.

`Model Training:` After the data preprocessing steps, the pipeline includes the training of a machine learning model. This can be a classifier for classification tasks, a regressor for regression tasks, or any other type of model depending on the problem at hand.

`Model Evaluation:` Once the model is trained, the pipeline often incorporates steps for evaluating its performance. This may involve metrics calculation, cross-validation, or any other evaluation technique to assess the model's effectiveness.

`Predictions:` After the model has been evaluated, the pipeline allows you to make predictions on new, unseen data using the trained model. This step applies the same preprocessing steps used during training to the new data before generating predictions.

The main advantages of using pipelines in machine learning are:

`Simplified Workflow:` Pipelines provide a clean and organized structure for defining and managing the sequence of steps involved in machine learning tasks. This makes it easier to understand, modify, and reproduce the workflow.

`Avoiding Data Leakage:` Pipelines ensure that data preprocessing steps are applied consistently to both the training and testing data, preventing data leakage that could lead to biased or incorrect results.

`Streamlined Model Deployment:` Pipelines allow you to encapsulate the entire workflow, including data preprocessing and model training, into a single object. This simplifies the deployment of your machine learning model, as the same pipeline can be applied to new data without the need to reapply each individual step.

`Hyperparameter Tuning:` Pipelines can be combined with techniques like grid search or randomized search for hyperparameter tuning. This allows you to efficiently explore different combinations of hyperparameters for your models.

Summary:

Overall, pipelines are a powerful tool for managing and automating the machine learning workflow, promoting code reusability, consistency, and efficiency. They help streamline the development and deployment of machine learning models, making it easier to iterate and experiment with different approaches.

```
[ ]: # import libraries
     import pandas as pd
     import seaborn as sns
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import accuracy_score
     from sklearn.pipeline import Pipeline
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.preprocessing import StandardScaler, OneHotEncoder
     from sklearn.impute import SimpleImputer
     from sklearn.compose import ColumnTransformer
```

```
[ ]: titanic = sns.load_dataset('titanic')

     # display first 5 rows
     titanic.head()
```

```
[ ]:    survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
     0         0       3    male  22.0      1      0   7.2500        S  Third
     1         1       1  female  38.0      1      0  71.2833        C  First
     2         1       3  female  26.0      0      0   7.9250        S  Third
     3         1       1  female  35.0      1      0  53.1000        S  First
     4         0       3    male  35.0      0      0   8.0500        S  Third

          who  adult_male deck  embark_town alive  alone
     0    man        True  NaN  Southampton    no  False
     1  woman       False    C    Cherbourg   yes  False
     2  woman       False  NaN  Southampton   yes   True
     3  woman       False    C  Southampton   yes  False
     4    man        True  NaN  Southampton    no   True
```

```
[ ]: # Select features and target variable
     X = titanic_data[['pclass', 'sex', 'age', 'fare', 'embarked']]
     y = titanic_data['survived']

     # Split the data into train and test sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
      ↪random_state=42)

     # Define the column transformer for imputing missing values
     numeric_features = ['age', 'fare']
     categorical_features = ['pclass', 'sex', 'embarked']
```

```
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median'))
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])

# Create a pipeline with the preprocessor and RandomForestClassifier
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(random_state=42))
])

# Fit the pipeline on the training data
pipeline.fit(X_train, y_train)

# Make predictions on the test data
y_pred = pipeline.predict(X_test)

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7821229050279329
```

2 Hyperparameter Tuning in Pipeline

[ ]:
```
from sklearn.model_selection import GridSearchCV

pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(handle_unknown='ignore')),
    ('model', RandomForestClassifier(random_state=42))
])

# Define the hyperparameters to tune
hyperparameters = {
    'model__n_estimators': [100, 200, 300, 500],
    'model__max_depth': [None, 5, 10, 30],
    'model__min_samples_split': [2, 5, 10, 15]
}

# Perform grid search cross-validation
grid_search = GridSearchCV(pipeline, hyperparameters, cv=5)
grid_search.fit(X_train, y_train)

# Get the best model
best_model = grid_search.best_estimator_

# Make predictions on the test data using the best model
y_pred = best_model.predict(X_test)

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Print the best hyperparameters
print("Best Hyperparameters:", grid_search.best_params_)
```

```
Accuracy: 0.8212290502793296
Best Hyperparameters: {'model__max_depth': 30, 'model__min_samples_split': 5,
'model__n_estimators': 100}
```

3 Explanation of the Pipeline and Model Training

1. **Data Preparation:**
    - The features (X) and target variable (y) are extracted from the Titanic dataset.
    - The data is split into training and testing sets using train_test_split.
2. **Feature Engineering:**
    - Numeric and categorical features are identified.
    - Separate pipelines are created for numeric and categorical features:
        - Numeric features: Imputed with mean values and scaled using StandardScaler.
        - Categorical features: Imputed with most frequent values and one-hot encoded.
3. **Pipeline Creation:**
    - A ColumnTransformer combines the numeric and categorical transformers.
    - The final pipeline includes the preprocessor and a RandomForestClassifier.
4. **Model Training and Evaluation:**
    - The pipeline is fitted on the training data.
    - Predictions are made on the test data.
    - The model's accuracy is calculated and printed.
5. **Hyperparameter Tuning:**
    - A parameter grid is defined for the RandomForestClassifier.
    - GridSearchCV is used to perform cross-validated search over the parameter grid.
    - The best parameters and cross-validation score are printed.
6. **Final Evaluation:**

- The best model from the grid search is used to make predictions on the test data.
- The final accuracy of the best model is calculated and printed.