WinePredict

Release 0.1.0

Chris Lawrence

Aug 18, 2024

CONTENTS:

CHAPTER

ONE

INTRODUCTION

WinePredict is a Python library designed to facilitate wine price prediction using advanced machine learning models. This library provides a comprehensive set of tools for data processing, model training, evaluation, and visualization, specifically tailored for the wine industry.

1.1 Key Features

1. **Data Processing**: Automated collection and preprocessing of economic indicators from FRED (Federal Reserve Economic Data).
2. **Multiple Models**: Implementation of various machine learning models including:
    - Linear Regression
    - Ridge Regression
    - Lasso Regression
    - Neural Networks
    - Support Vector Machines
    - Decision Trees
    - Random Forests
    - Gradient Boosting
    - XGBoost
    - LightGBM
    - CatBoost
3. **Model Evaluation**: Comprehensive evaluation metrics including R2, Adjusted R2, RMSE, and MAE.
4. **Feature Importance**: Analysis and visualization of the most influential factors in wine price prediction.
5. **Visualization Tools**: Functions for plotting actual vs. predicted prices, residual analysis, and more.

1.2 Target Audience

WinePredict is designed for:

- Data scientists and analysts in the wine industry
- Wine producers and industry professionals (brokers, importers, etc) seeking data-driven insights

This documentation provides an overview of the library's features, installation instructions, usage examples, and a complete API reference to help you get started with WinePredict.

CHAPTER

TWO

USAGE

This section provides examples of how to use the WinePredict library for wine price prediction.

2.1 Basic Usage

Here's a simple example of how to use WinePredict to download data, preprocess it, train models, and evaluate their performance:

```
from winepredict.data_processing import preprocess_and_analyze_data,
    process_fred_data
from winepredict.model_training import train_and_evaluate_models,
    tune_and_evaluate_catboost
from winepredict.model_evaluation import evaluate_model, cross_validate_model
from winepredict.visualization import plot_actual_vs_predicted,
    visualize_residuals, plot_feature_importance

# Define FRED API key
api_key = 'your_api_key_here'

# Download and process FRED data
process_fred_data(api_key)

# Preprocess and analyze data
file_path = 'FRED_Data.xlsx'
scaled_df = preprocess_and_analyze_data(file_path, output_file='
    correlation_matrix.png', save_vif=True)
```

```

# Train and evaluate multiple models
results_df = train_and_evaluate_models(scaled_df)

# Tune and evaluate CatBoost model
X = scaled_df.drop(['Average Wine Price'], axis=1)
y = scaled_df['Average Wine Price']
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8,
    shuffle=True, random_state=42)
catboost_results = tune_and_evaluate_catboost(X_train, y_train, X_test, y_test
    )
best_catboost = catboost_results['best_estimator']
```

```
# Visualize results
plot_actual_vs_predicted(y_test, best_catboost.predict(X_test))
visualize_residuals(y_test, best_catboost.predict(X_test))
plot_feature_importance(catboost_results['ranked_feature_importance'])
```

This example demonstrates the basic workflow of using WinePredict, from data processing to model evaluation and visualization.

2.2 Advanced Usage

For more advanced usage and customization options, please refer to the API Reference section.

CHAPTER

THREE

DATA PROCESSING

The `data_processing` module is responsible for preparing and processing data for wine price prediction analysis. It handles data retrieval from FRED (Federal Reserve Economic Data), data cleaning, preprocessing, and initial analysis.

3.1 Module Contents

```
import pandas as pd
import numpy as np
import requests
from datetime import datetime
import openpyxl
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

3.1.1 Functions Overview

The `data_processing` module contains the following key functions:

1. **download_fred_data**: Downloads data from FRED for a given series ID.
2. **process_fred_data**: Processes FRED data by downloading, filtering, resampling, and saving it to an Excel file.
3. **calculate_vif**: Calculates the Variance Inflation Factor (VIF) for each feature in the dataset.
4. **preprocess_and_analyze_data**: Loads, preprocesses, and performs correlation analysis on the data.

3.1.2 Detailed Function Descriptions

download_fred_data

```
def download_fred_data(series_id, api_key):
```

Description: Downloads data from FRED for a given series ID.

Parameters:

- `series_id` (str): The FRED series ID.
- `api_key` (str): The API key for accessing FRED data.

Returns:

- pandas.DataFrame: A DataFrame containing the downloaded data with dates as the index.

process_fred_data

```
def process_fred_data(api_key, start_date='1992-01-01', output_file='FRED_Data.xlsx'):
```

Description: Processes FRED data by downloading, filtering, resampling, and saving it to an Excel file.

Parameters:

- `api_key` (str): The API key for accessing FRED data.
- `start_date` (str): The start date for filtering the data. Defaults to '1992-01-01'.
- `output_file` (str): The name of the output Excel file. Defaults to 'FRED_Data.xlsx'.

calculate_vif

```
def calculate_vif(df):
```

Description: Calculates VIF for each feature in the DataFrame.

Parameters:

- `df` (pandas.DataFrame): The DataFrame containing the features.

Returns:

- pandas.DataFrame: A DataFrame containing the VIF values for each feature.

preprocess_and_analyze_data

```
def preprocess_and_analyze_data(file_path, sheet_name='Data', output_file='correlation_matrix.png', save_vif=False):
```

Description: Loads, preprocesses, and performs correlation analysis on the data.

Parameters:

- `file_path` (str): The path to the Excel file containing the data.
- `sheet_name` (str): The name of the sheet in the Excel file to load. Defaults to 'Data'.
- `output_file` (str): The file path to save the correlation matrix image. Defaults to 'correlation_matrix.png'.
- `save_vif` (bool): Whether to save VIF results to an Excel file. Defaults to False.

Returns:

- pandas.DataFrame: A DataFrame containing the scaled data with the 'Average Wine Price' column.

3.1.3 Usage Example

Here's an example of how to use the functions in the `data_processing` module:

```
from winepredict.data_processing import process_fred_data,
    preprocess_and_analyze_data

# Process FRED data
api_key = "your_fred_api_key"
process_fred_data(api_key, start_date='2000-01-01', output_file='Wine_Data.
    xlsx')

# Preprocess and analyze the data
scaled_df = preprocess_and_analyze_data('Wine_Data.xlsx', save_vif=True)

print(scaled_df.head())
```

This example demonstrates downloading FRED data, preprocessing it, and performing initial analysis for wine price prediction.

CHAPTER

FOUR

MODEL TRAINING

The `model_training` module is responsible for training, evaluating, and fine-tuning various machine learning models used in the WinePredict library. It provides functions for comparing multiple regression models and optimizing the CatBoost model for wine price prediction.

4.1 Module Contents

This module imports several regression models and utility functions:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor
from catboost import CatBoostRegressor
from sklearn.metrics import R2_score, mean_squared_error, mean_absolute_error
from sklearn.exceptions import ConvergenceWarning
import matplotlib.pyplot as plt
import warnings
```

4.1.1 Functions Overview

The `model_training` module contains two main functions:

1. **train_and_evaluate_models**: Trains multiple regression models and evaluates their performance.
2. **tune_and_evaluate_catboost**: Performs hyperparameter tuning for the CatBoost model and evaluates its performance.

4.1.2 Detailed Function Descriptions

train_and_evaluate_models

```
def train_and_evaluate_models(scaled_df):
```

Description: This function splits the dataset, trains multiple regression models, and evaluates their performance using various metrics.

Models included:

- Linear Regression
- Ridge Regression
- Lasso Regression
- Neural Network (MLPRegressor)
- Support Vector Machine (SVR)
- Decision Tree
- Random Forest
- Gradient Boosting
- XGBoost
- LightGBM
- CatBoost

Parameters:

- `scaled_df` (pd.DataFrame): The preprocessed and scaled DataFrame containing the features and target variable.

Returns:

- pd.DataFrame: A DataFrame containing the performance metrics (R2, Adjusted R2, RMSE, MAE) of each model.

tune_and_evaluate_catboost

```
def tune_and_evaluate_catboost(X_train, y_train, X_test, y_test):
```

Description: This function performs hyperparameter tuning for the CatBoost model using GridSearchCV, evaluates the best model, and visualizes feature importance.

Parameters:

- `X_train` (pd.DataFrame): Training features.
- `y_train` (pd.Series): Training target variable.
- `X_test` (pd.DataFrame): Testing features.
- `y_test` (pd.Series): Testing target variable.

Returns:

- dict: A dictionary containing the best parameters, performance metrics (R2, RMSE, MAE), and ranked feature importance of the best CatBoost model.

4.1.3 Usage Example

Here's an example of how to use the functions in the `model_training` module:

```
from winepredict.model_training import train_and_evaluate_models,
    tune_and_evaluate_catboost
from winepredict.data_processing import preprocess_and_analyze_data

# Assume we have already processed and scaled our data
scaled_df = preprocess_and_analyze_data('Wine_Data.xlsx')

# Train and evaluate multiple models
results_df = train_and_evaluate_models(scaled_df)
print(results_df)

# Split the data for CatBoost tuning
X = scaled_df.drop(['Average Wine Price'], axis=1)
y = scaled_df['Average Wine Price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

# Tune and evaluate CatBoost
catboost_results = tune_and_evaluate_catboost(X_train, y_train, X_test, y_test
    )
print("Best CatBoost R2:", catboost_results['R2'])
print("Top 5 important features:", catboost_results['ranked_feature_importance
    '].head())
```

4.2 Classes Overview

The following classes are available in the `model_training` module. They represent different machine learning models and utilities:

1. **CatBoostRegressor**: A regressor based on the CatBoost algorithm, which is particularly effective with categorical features and small datasets.
2. **ConvergenceWarning**: A warning class used when a model does not converge during training.
3. **DecisionTreeRegressor**: A decision tree regressor that can be used for predicting continuous values.
4. **GradientBoostingRegressor**: A regressor based on the gradient boosting algorithm, combining multiple weak models into a strong one.
5. **GridSearchCV**: A class for exhaustive search over specified hyperparameters for an estimator.
6. **KNeighborsRegressor**: A regressor that uses the k-nearest neighbors algorithm (optional).
7. **LGBMRegressor**: A LightGBM regressor, known for its speed and efficiency in large datasets.
8. **Lasso**: A linear model with L1 regularization, useful for feature selection.
9. **LinearRegression**: A basic linear regression model for predicting continuous values.
10. **MLPRegressor**: A multi-layer perceptron regressor, suitable for complex, non-linear problems.
11. **RandomForestRegressor**: A regressor that fits multiple decision trees on various sub-samples of the dataset.
12. **Ridge**: A linear regression model with L2 regularization to prevent overfitting.

13. **SVR**: A support vector regressor for robust, non-linear predictions.
14. **XGBRegressor**: A regressor based on the XGBoost algorithm, effective for tabular data.

4.3 Imported Models

These models are imported from various libraries including scikit-learn, XGBoost, LightGBM, and CatBoost.

4.3.1 Utility Classes and Functions

- **train_test_split**: Function for splitting datasets into training and testing subsets.
- **GridSearchCV**: Utility for performing grid search with cross-validation for hyperparameter tuning.
- **r2_score, mean_squared_error, mean_absolute_error**: Metric functions for evaluating model performance.
- **ConvergenceWarning**: Warning class used to handle convergence issues in iterative algorithms.

4.3.2 Usage in the Module

The imported models are used in the following ways:

1. In the `train_and_evaluate_models` function, each regression model is instantiated, trained on the training data, and evaluated on the test data.
2. The `CatBoostRegressor` is additionally used in the `tune_and_evaluate_catboost` function for hyperparameter tuning and detailed evaluation.
3. Utility functions like `train_test_split` and `GridSearchCV` are used for data preparation and model optimization.
4. Metric functions are used to calculate performance metrics such as R^2, RMSE, and MAE for each model.

4.3.3 Note on Documentation

For detailed documentation on these imported classes and functions, please refer to their respective library documentation:

- Scikit-learn: https://scikit-learn.org/stable/documentation.html
- XGBoost: https://xgboost.readthedocs.io/
- LightGBM: https://lightgbm.readthedocs.io/
- CatBoost: https://catboost.ai/docs/

4.4 Functions Overview

The `model_training` module includes the following key functions:

1. **train_and_evaluate_models**: Trains and evaluates a set of machine learning models, comparing their performance using cross-validation and other metrics.
2. **tune_and_evaluate_catboost**: Tunes hyperparameters for the CatBoostRegressor using grid search and evaluates its performance.

4.5 Detailed Function Descriptions

The `model_training` module contains two key functions that form the core of its functionality. Here are detailed descriptions of these functions:

```
def train_and_evaluate_models(scaled_df):
    """Splits the dataset, trains multiple models, and evaluates their
       performance."""

def tune_and_evaluate_catboost(X_train, y_train, X_test, y_test):
    """Tunes hyperparameters for CatBoost, evaluates the best model, and
       visualizes feature importance."""
```

Description:

- Trains multiple machine learning models and evaluates them using cross-validation.
- This function provides a comprehensive comparison of models' performance across different metrics.

Parameters:

- `X` (pandas.DataFrame): The feature matrix.
- `y` (pandas.Series): The target variable.
- `models` (list): A list of machine learning models to train.
- `cv` (int): The number of cross-validation folds (default: 5).

Returns:

- dict: A dictionary containing the performance metrics for each model.

Description:

- Performs hyperparameter tuning for the CatBoostRegressor and evaluates its performance on the validation set.
- This function helps in finding the best set of hyperparameters for the CatBoost model.

Parameters:

- `X_train` (pandas.DataFrame): The training feature matrix.
- `y_train` (pandas.Series): The training target variable.
- `X_val` (pandas.DataFrame): The validation feature matrix.
- `y_val` (pandas.Series): The validation target variable.

Returns:

- object: The tuned CatBoostRegressor model.

4.6 Usage Examples

Here's an example of how to use the functions in the `model_training` module:

```
from winepredict.model_training import train_and_evaluate_models,
    tune_and_evaluate_catboost
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.model_selection import train_test_split

# Sample data
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
    random_state=42)

# Initialize models
models = [RandomForestRegressor(), GradientBoostingRegressor()]

# Train and evaluate models
results = train_and_evaluate_models(X_train, y_train, models, cv=5)
print("Model evaluation results:", results)

# Tune and evaluate CatBoost
best_catboost = tune_and_evaluate_catboost(X_train, y_train, X_val, y_val)
```

CHAPTER

FIVE

MODEL EVALUATION

The `model_evaluation` module offers tools for assessing the performance of trained wine price prediction models. It includes functions for:

- Cross-validation
- Computation of performance metrics
- Model comparison and analysis

These utilities enable users to gauge model generalization, identify areas for improvement, and select the most effective models for wine price prediction.

5.1 Module Contents

```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from sklearn.model_selection import cross_val_score
import numpy as np
```

The module primarily utilizes scikit-learn's metrics and model selection tools, along with NumPy for numerical operations.

5.2 Functions Overview

The following functions are part of the `model_evaluation` module:

1. **cross_validate_model**: Performs cross-validation on the model, providing a robust estimate of its performance on unseen data by splitting the data into multiple folds.
2. **evaluate_model**: Evaluates a trained model using various performance metrics, such as accuracy, precision, recall, and F1 score.

5.3 Detailed Function Descriptions

Below are detailed descriptions of each function within the `model_evaluation` module:

```
winepredict.model_evaluation.cross_validate_model
```

Description:

- Performs k-fold cross-validation on the specified model and dataset.
- This function is useful for assessing the model's performance and variability across different subsets of data.

Parameters:

- `model` (object): The machine learning model to be evaluated.
- `X` (pandas.DataFrame): The feature matrix.
- `y` (pandas.Series): The target variable.
- `cv` (int): The number of cross-validation folds (default: 5).

Returns:

- dict: A dictionary containing cross-validation scores for each fold, along with the mean and standard deviation.

```
winepredict.model_evaluation.evaluate_model
```

Description:

- Evaluates a trained model's performance using multiple metrics.
- This function can be used to calculate accuracy, precision, recall, F1 score, and confusion matrix, among others.

Parameters:

- `model` (object): The trained machine learning model.
- `X_test` (pandas.DataFrame): The test feature matrix.
- `y_test` (pandas.Series): The true labels for the test set.

Returns:

- dict: A dictionary containing the calculated performance metrics.

5.4 Usage Examples

The following example demonstrates how to use the functions in the `model_evaluation` module:

```
from winepredict.model_evaluation import cross_validate_model, evaluate_model
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# Sample data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)

# Initialize a model
model = RandomForestClassifier()
```

```
10
11 # Perform cross-validation
12 cv_results = cross_validate_model(model, X_train, y_train, cv=5)
13 print("Cross-validation results:", cv_results)
14
15 # Train the model
16 model.fit(X_train, y_train)
17
18 # Evaluate the model
19 evaluation_metrics = evaluate_model(model, X_test, y_test)
20 print("Evaluation metrics:", evaluation_metrics)
```

CHAPTER

SIX

VISUALIZATION

The `visualization` module provides functions for visualizing the results and insights from the WinePredict library. These visualizations include plotting actual vs. predicted values, analyzing feature importance, and examining residuals. These tools are essential for understanding model performance and identifying areas for improvement.

6.1 Module Contents

```
winepredict.visualization
```

6.2 Functions Overview

The `visualization` module includes the following key functions:

1. **plot_actual_vs_predicted**: Generates a plot comparing the actual vs. predicted values from a regression model, allowing for quick visual assessment of model performance.
2. **plot_feature_importance**: Visualizes the importance of each feature used by the model, aiding in understanding which features contribute most to the predictions.
3. **visualize_residuals**: Plots the residuals of the model predictions, helping to diagnose issues such as heteroscedasticity or non-linearity.

6.3 Detailed Function Descriptions

```
winepredict.visualization.plot_actual_vs_predicted
```

Description:

- Generates a scatter plot comparing the actual target values with the model's predicted values.
- This plot helps to identify how well the model is performing, highlighting areas where the model may be under or over-predicting.

Parameters:

- `y_true` (array-like): The actual target values.
- `y_pred` (array-like): The predicted values from the model.
- `title` (str, optional): The title of the plot.

- `save_path` (str, optional): Path to save the plot image.

```
winepredict.visualization.plot_feature_importance
```

Description:

- Creates a bar plot showing the importance of each feature in the model.
- Useful for feature selection and understanding model behavior.

Parameters:

- `model` (object): A trained model with a `feature_importances_` attribute.
- `feature_names` (list of str): The names of the features.
- `title` (str, optional): The title of the plot.
- `save_path` (str, optional): Path to save the plot image.

```
winepredict.visualization.visualize_residuals
```

Description:

- Generates a residuals plot, which is the difference between the actual and predicted values.
- Helps in diagnosing model issues such as non-linearity, outliers, or patterns in the residuals that suggest a problem with the model fit.

Parameters:

- `y_true` (array-like): The actual target values.
- `y_pred` (array-like): The predicted values from the model.
- `title` (str, optional): The title of the plot.
- `save_path` (str, optional): Path to save the plot image.

CHAPTER

SEVEN

API REFERENCE

This section provides detailed information about the modules and functions in the WinePredict library.

7.1 Overview

The WinePredict library is a comprehensive toolkit designed for the analysis, training, evaluation, and visualization of models related to wine data prediction. This documentation covers the primary modules and key functions that are essential for working with the library.

7.2 Modules Overview

The library is divided into several core modules:

- **Data Processing**: Prepares and processes data for analysis and modeling.
- **Model Training**: Handles the training and tuning of predictive models.
- **Model Evaluation**: Evaluates the performance of trained models using various metrics.
- **Visualization**: Provides tools for visualizing data, predictions, and model performance.

7.3 Data Processing Module

The Data Processing module contains functions that load, clean, and prepare data for analysis and model training.

```
winepredict.data_processing
```

7.3.1 Key Functions

- **process_fred_data**: Loads and processes data from the FRED API for analysis.
- **preprocess_and_analyze_data**: Performs preprocessing and exploratory analysis on the data.

7.4 Model Training Module

The Model Training module contains functions that train machine learning models and tune their hyperparameters for optimal performance.

```
winepredict.model_training
```

7.4.1 Key Functions

- **train_and_evaluate_models**: Trains various models and evaluates their performance.
- **tune_and_evaluate_catboost**: Tunes and evaluates the CatBoost model for better accuracy.

7.5 Model Evaluation Module

The Model Evaluation module provides tools for assessing the performance of the trained models using cross-validation and other metrics.

```
winepredict.model_evaluation
```

7.5.1 Key Functions

- **evaluate_model**: Evaluates a model's performance on a given test dataset.
- **cross_validate_model**: Performs cross-validation to assess model stability and performance.

7.6 Visualization Module

The Visualization module offers functions for plotting various aspects of the data and model predictions.

```
winepredict.visualization
```

7.6.1 Key Functions

- **plot_actual_vs_predicted**: Plots the actual vs. predicted values for a model.
- **visualize_residuals**: Visualizes residuals to assess the model's accuracy.
- **plot_feature_importance**: Displays feature importance as calculated by the model.

7.7 Usage Examples

For more details on how to use these functions, refer to the following sections:

- **Data Processing**: Loading and cleaning data before modeling.
- **Model Training**: Training and tuning models.
- **Model Evaluation**: Evaluating model performance.
- **Visualization**: Creating plots for model results.

Refer to the module-specific sections for comprehensive details on all available functions and classes.