Final project

Presented By: Godsway Akakpo

BAN6800 Business Analytics Capstone

Lecturer**: Raphael Wanjiku**

Submission Date: 2nd October 2024

<u>Documentation</u>

i. **Overview of the script**

This Python script is designed for product pricing optimization using machine learning models. It handles the following key tasks:

1. Data preprocessing: Handling missing values and performing feature engineering.

2. Model training and evaluation: Using ElasticNet, Random Forest, XGBoost, and Stacking models.

3. Model optimization: Hyperparameter tuning using GridSearchCV and cross-validation.

4. Price optimization: Predicting demand based on different prices and calculating optimal pricing for maximum revenue.

5. Visualization and reporting: Generating plots to visualize results.

6. Saving the model: The final model is saved for future use.

ii. Libraries and Modules

- pandas, numpy: this library is used for data manipulation and numerical computations.

- scikit-learn: Provides machine learning models, pipelines, preprocessing, model evaluation, and hyperparameter tuning.

- xgboost: An advanced gradient boosting library used for building the XGBoost model.

- matplotlib, seaborn: used for data visualization.

- joblib: This is used to save the trained models for future use.

iii. Key Functions

1. `handle_missing_values(df)`

This function handles missing values by imputing the median for numeric columns and the mode for categorical columns. And returns;

A DataFrame with no missing values.

2. `feature_engineering(df)`

This function is used to create new features based on interaction and polynomial terms (e.g., interaction between price and marketing spend, and square of price). It returns;

The DataFrame with additional features for improved model performance.

3. `preprocess_data(df, target='Demand_Units_Sold')`

This function also combines the steps of missing value handling and feature engineering. It splits the data into features (X) and target (y) for model training.

It returns;

- `X`: The feature matrix.

- `y`: The target variable (demand).

4. `plot_corr_heatmap(df)`

Generates a heatmap of the correlations between numeric features to identify relationships in the dataset.

This helps display the correlation heatmap.

5. `build_preprocessor()`

This helps build a preprocessing pipeline that scales numeric features and one-hot encodes categorical features using `ColumnTransformer` and returns;

A `preprocessor` object for use in model pipelines.

6. `evaluate_model(model, X_test, y_test, model_name)`

This function helps evaluate the model's performance using metrics like R^2 and RMSE on the test dataset.

It returns;

- `r2`: R-squared value (indicates goodness of fit).

- `rmse`: Root Mean Squared Error (indicates error magnitude).

7. `plot_actual_vs_predicted(y_test, y_pred, model_name)`

Plots the actual versus predicted values for visual comparison of model performance.

It displays the output in the form of a scatter plot comparing actual and predicted values.

8. `hyperparameter_tuning(model, param_grid, X_train, y_train, model_name)`

This function uses GridSearchCV for hyperparameter tuning to find the best parameters for each model.

This helps return the best model with the optimal hyperparameters.

9. `optimize_price(model, base_price, competitor_price, customer_ratings, marketing_spend, preprocessor)`

This function helps find the optimal price by predicting demand for a range of prices and calculating the corresponding revenue.

It returns the optimal price for maximizing revenue.

10. `main()`

The main pipeline that integrates all the steps:

1. Data preprocessing.
2. Training multiple models (ElasticNet, RandomForest, XGBoost, and Stacking).
3. Evaluating the models.
4. Price optimization.
5. Saving the final stacking model using `joblib`.

iv. Modeling Details

- ElasticNet: A linear model that combines Lasso and Ridge regularization. It helps with feature selection and reducing overfitting.
- Random Forest: A tree-based ensemble model that reduces variance by averaging multiple decision trees.

- XGBoost: An efficient implementation of gradient boosting, widely used for structured data problems.

- Stacking Model: Combines the predictions of ElasticNet, RandomForest, and XGBoost using a meta-model (ElasticNet) for enhanced performance.

Hyperparameter Tuning

Hyperparameter tuning is performed for ElasticNet and RandomForest using GridSearchCV with cross-validation. This ensures that the model's hyperparameters are optimized for better performance.

V. Visualization

The script generates the following plots for insights:

- Correlation Heatmap: Shows relationships between numeric features.

- Actual vs Predicted Plot: Visual comparison of actual and predicted values for the stacking model.

VI. Price Optimization

The script evaluates a range of prices and computes the corresponding demand and revenue. It outputs the optimal price that maximizes revenue based on the model's predictions.

Saving and Loading the Model

- The trained stacking model is saved as a `pkl` file using joblib for future use without retraining.

- To load and use the model later, you can use:

```python
model = joblib.load('stacking_pricing_model.pkl')
```