Data Science Project Protocol

Bar-Ilan University

Author:

Nadav Marciano

Introduction

London, the capital of the United Kingdom, is one of the world's most diverse and vibrant metropolitan centers. The city is renowned for its culture, commerce, education, fashion, and gastronomy. Among its many culinary scenes, Indian cuisine has become particularly prominent, reflecting both historical connections with India and the city's multicultural population. Today, Indian restaurants are among the most popular dining choices in London, serving a wide range of traditional and innovative dishes.

In recent years, food delivery services have grown rapidly, driving restaurants to adapt their menus and operations to meet customer expectations. This shift has created new challenges: restaurants must manage their inventory efficiently to ensure that ingredients remain fresh while minimizing waste. Predicting customer demand has become increasingly complex due to various external factors such as weather conditions, holidays, and salary cycles at the beginning of the month.

This project focuses on delivery orders from an Indian restaurant in London. The research question is straightforward yet highly practical: which products should the restaurant stock for the upcoming three days in order to meet delivery demand? By answering this question, the restaurant can optimize its supply chain, reduce waste, and provide customers with consistently fresh meals. Ultimately, the study aims to demonstrate how data-driven methods can support better decision-making in the food delivery industry.

Methodology

Data Sources:

The dataset used in this project was collected from *Kaggle* and contains delivery records from an Indian restaurant in London. It spans a period of three years and includes over 300 unique products, together with detailed information on orders and prices.

To enrich the data, additional sources were integrated:

- Calendar data: for each date, information was added regarding the day of the week, the month, and whether the date coincided with a public holiday.

- Weather data: daily weather statistics in London, including temperature, precipitation, humidity, dew point, and wind speed. Each variable was represented by its minimum, maximum, and average values.

The overall project timeline was approximately two months. The main effort was dedicated to data collection and preparation, including cleaning and integration. Model training was allocated about one week, and model evaluation was scheduled for three days.

Data Preparation and Target Variables:

The initial stage of data processing was carried out in Python as well as in SQL, where a **flat-file table** was constructed. Before building the main dataset, all restaurant variables were converted to lowercase, and duplicate product entries were removed.

Next, the ***Apriori algorithm*** was applied to the data in order to identify the most frequent product combinations. These combinations were then used to define the project's **target variables**.

Specifically, three different target variables ("meals") were created, each representing sets of products that are frequently ordered together:

1. The **first target variable** was based on the three most frequent items in the restaurant's dataset.
2. The **second target variable** was constructed using the *Apriori algorithm*, based on product combinations with the highest *lift* values.
3. The **third target variable** was also derived from frequent product associations, capturing additional significant combinations.

Each target variable served as the foundation for building three separate datasets, which were later used to train and evaluate different models.

Target Encoding and Exploratory Data Analysis (EDA):

For each of the three target variables defined earlier, a **binary categorical encoding** was applied:

- **1** – If the specific meal (i.e., the product combination) was ordered.
- **0** – If the meal was not ordered.

In order to forecast demand for the **upcoming three days**, a three-day **lag** was introduced. This allowed the model to predict whether these product combinations would be ordered in the following days, ensuring that the restaurant could check stock availability and place orders with suppliers in advance.

The dataset was then divided into three separate data frames (numeric, categorical, and target variables) to facilitate the **Exploratory Data Analysis (EDA)** process. Statistical properties of each variable were examined, including distributions and correlations with the target variables. Strong correlations and significant relationships were identified and compared against the targets to determine relevance.

Potential **outliers** were also investigated. Each outlier was checked for its impact on correlations and distributions. If the presence of outliers significantly distorted relationships, they were removed. Ultimately, outlier values were excluded from all three datasets to improve model robustness.

	var	outliers_pct	distribution_change	correlation_change	drop
0	Total_price	33	-	-	yes
4	min_temperature	27	-	-	yes
21	max_cloud_cover	27	-	-	yes
10	min_dew_point	22	-	-	yes
8	max_dew_point	20	-	-	yes
9	avg_dew_point	17	-	-	yes
3	avg_temperature	14	-	-	yes
2	max_temperature	3	-	-	yes
20	precipitation	2	-	-	yes
1	Order_count	1	-	-	yes
13	min_wind_speed	1	-	-	yes
16	min_pressure_sea	0	-	-	yes
22	avg_cloud_cover	0	-	-	yes
19	min_visibility	0	-	-	yes
18	avg_visibility	0	-	-	yes
17	max_visibility	0	-	-	yes
12	avg_wind_speed	0	-	-	yes
15	avg_pressure_sea	0	-	-	yes
14	max_pressure_sea	0	-	-	yes
11	max_wind_speed	0	-	-	yes
7	min_relative_humidity	0	-	-	yes
6	avg_relative_humidity	0	-	-	yes
5	max_relative_humidity	0	-	-	yes
23	min_cloud_cover	0	-	-	yes

Table 1 – EDA: Outlier Detection and Variable Exclusion Criteria

Table 1 summarizes the percentage of outliers detected for each variable, indicates whether they affected distribution or correlation, and shows which variables were excluded from further analysis.

Handling Missing Values:

The dataset contained 11 variables with missing values. Missing data were addressed at both the row and column levels:

- **Row-wise**: records with more than 50% missing values were removed.
- **Column-wise**: in cases where the proportion of missing values was relatively small, the variables were retained. Each affected variable was examined to ensure that the missing values did not significantly alter its distribution or its correlation with other variables.

	missing	var	missing_cnt	distribution_change
0	Total_price	Date	33	-
1	Total_price	Order_count	33	-
2	Total_price	max_temperature	33	-
3	Total_price	avg_temperature	33	-
4	Total_price	min_temperature	33	-

Table 2 – EDA: Missing Values Analysis

Table 2 shows the variables with missing values, the number of missing records for each, and whether the missing data affected their distribution.

To address the remaining missing values, a **K-Nearest Neighbors (KNN)** imputation model was applied. This method was performed consistently across all three datasets to ensure completeness and maintain the reliability of the features. The imputation process did not significantly alter the data distributions.

Below are two examples demonstrating the imputation process:

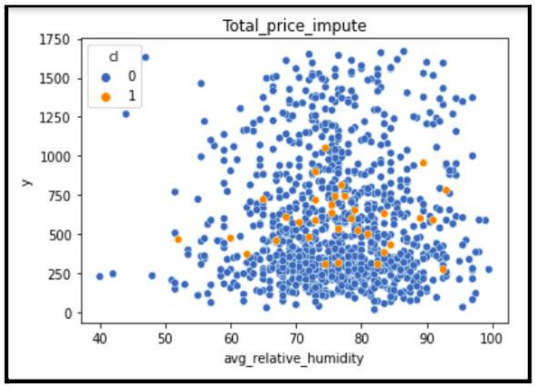

Figure 1 – KNN Imputation of Total Price

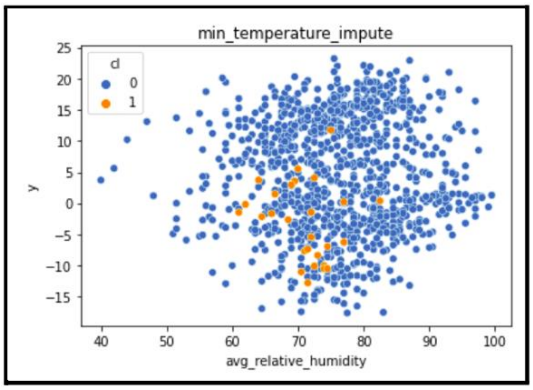

Figure 2 – KNN Imputation of Min Temperature

Models

Feature Engineering:

As a first step, **feature engineering** was performed on the dataset. All categorical variables were transformed into dummy variables according to the number of unique values in each category. In addition, based on the exploratory data analysis (EDA), new statistical features and ratios were created to enrich the dataset and improve the predictive power of the models.

Feature Selection:

A feature selection strategy was applied in order to reduce dimensionality and retain only the most relevant variables. Instead of training the models on all available features—which could introduce noise, redundancy, or risk of overfitting—several classification models were first evaluated to determine which variables consistently contributed most to predictive performance.

Each model highlights important features in different ways (e.g., decision-tree–based models rank features by information gain, while linear models assess their contribution through coefficients). By comparing the outputs of multiple models, it was possible to identify features that were repeatedly ranked as significant across different approaches. These variables were considered more robust and reliable predictors.

As a result, for each dataset, a **reduced feature set** was created containing only the selected variables. This ensured that subsequent models were trained on features that maximized predictive power while minimizing irrelevant information. Ultimately, this step improved both model efficiency and interpretability.

Data Splitting:

The data was then divided into training, validation, and test sets using the following proportions:

- **Train**: 60%
- **Validation**: 20%
- **Test**: 20%

This split ensured that each model could be trained, tuned, and evaluated in a structured and consistent

Target Variable 1 – Feature Set and Distribution:

For the first target variable, **40 features** were selected based on the feature selection process. These features were considered the most relevant for predicting whether the most frequent product combinations (“meals”) would be ordered.

	Variable	Univarable	Lasso	LogisticRegression	RandomForest	GradientBoost	SVM	Sum
0	Total_price	1	1	1	1	1	1	6
1	Order_count	1	0	1	1	1	1	5
4	min_temperature	0	0	1	1	1	1	4
5	max_relative_humidity	0	0	1	1	1	1	4
7	min_relative_humidity	0	0	1	1	1	1	4
14	max_pressure_sea	0	0	1	1	1	1	4
16	min_pressure_sea	0	0	1	1	1	1	4
20	precipitation	0	0	1	1	1	1	4
27	bombay aloo	1	0	1	1	1	0	4
46	crispy aloo	1	0	1	1	1	0	4
48	curry	1	0	1	1	1	0	4
49	garlic naan	1	0	1	1	1	0	4
50	Products2	1	0	1	1	1	1	5
51	korma	1	0	1	1	1	0	4
58	plain rice	1	0	1	1	0	1	4
69	pilau rice	1	0	1	1	1	0	4
79	week	1	0	1	1	1	1	5
80	weekday	0	0	1	1	1	1	4
96	keema naan	1	0	1	1	0	1	4
118	dhansak	1	0	1	1	1	0	4
171	mixed raita	1	0	1	1	1	0	4
188	red sauce	1	0	1	1	0	1	4
189	mint sauce	1	0	1	1	1	0	4

Table 3 – Selected Features for Target Variable 1

This table presents a subset of the most relevant features selected for Target Variable 1. For clarity, only a portion of the 40 engineered variables is shown.

To validate the quality of the dataset split, the distribution of the target variable was examined across the training, validation, and test sets. The results confirmed that the division was balanced and did not bias the model.

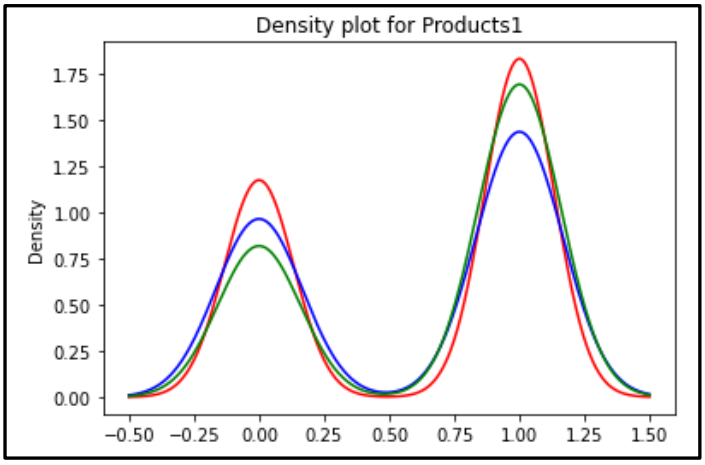

Figure 3 – Distribution of Target Variable 1 across Train, Validation, and Test Sets

This figure shows that the target variable is well distributed, ensuring a fair evaluation of the model.

Model Evaluation – Target Variable 1:

In the first stage, six different models were trained on the training set. Their performance was assessed using multiple evaluation metrics, including AUC, Accuracy, Log-loss, Precision, Recall, and F1-score. The results for each model on both the training and validation datasets are presented in Tables 4 and 5.

	model	Accuracy	Precision	Recall	f1-score	Log-loss	AUC
1	Decision Tree	1.000000	1.000000	1.000000	1.000000	2.220446e-16	1.000000
2	Random Forest	1.000000	1.000000	1.000000	1.000000	2.220446e-16	1.000000
4	GBM	0.926647	0.934466	0.945946	0.940171	2.643921e+00	0.921249
3	ADABoost	0.764970	0.804878	0.810811	0.807834	8.471338e+00	0.752149
0	Logistic Regression	0.717066	0.740088	0.825553	0.780488	1.019798e+01	0.686723
5	SVM	0.654192	0.673913	0.837838	0.746988	1.246420e+01	0.602827

Table 4 – Model Performance on Training Set

	model	Accuracy	Precision	Recall	f1-score	Log-loss	AUC
3	ADABoost	0.754464	0.816000	0.761194	0.787645	8.850004	0.752819
2	Random Forest	0.745536	0.803150	0.761194	0.781609	9.171823	0.741708
4	GBM	0.736607	0.815126	0.723881	0.766798	9.493641	0.739718
1	Decision Tree	0.674107	0.736434	0.708955	0.722433	11.746369	0.665589
0	Logistic Regression	0.674107	0.710345	0.768657	0.738351	11.746369	0.650995
5	SVM	0.633929	0.656627	0.813433	0.726667	13.194552	0.590050

Table 5 – Model Performance on Validation Set

From the comparison, AdaBoost emerged as the most suitable model. Unlike Decision Trees and Random Forests, which showed signs of overfitting with perfect training scores but weaker validation performance, AdaBoost achieved a more balanced performance between the training and validation sets. To further improve this model, hyper parameter fine-tuning was performed.

The tuning process explored key parameters such as `n_estimators`, `learning_rate`, and the depth and structure of the weak learners (`estimator__max_depth`, `estimator__min_samples_split`, `estimator__min_samples_leaf`). This was optimized through cross-validation on the training set, ensuring that the selected configuration generalized better to unseen data.

The best configuration identified by Grid Search was:

- `n_estimators = 300`
- `learning_rate = 0.05`
- `estimator__max_depth = 2`
- `estimator__min_samples_split = 2`
- `estimator__min_samples_leaf = 1`

With these optimized hyper parameters, the AdaBoost model improved by +5.47% AUC on the validation set compared to the baseline model. When applied to the independent test set, the tuned model maintained consistent performance (AUC = 0.798), confirming its robustness and ability to generalize.

Summary:

In summary, the evaluation of six classification models showed that AdaBoost provided the most promising balance between training and validation performance. Through careful hyper parameter optimization, the model achieved a significant improvement in predictive accuracy and maintained consistent results on the test set. This demonstrates the effectiveness of ensemble-based approaches, particularly when combined with fine-tuning, in addressing potential overfitting issues and enhancing generalization.

The same modeling pipeline applied to Target Variable 1 was also implemented for Target Variables 2 and 3. Specifically, the process included exploratory data analysis, feature engineering, feature selection, model training with six different classifiers, and subsequent hyper parameter fine-tuning of the most promising model (AdaBoost).

Since the procedure was identical to that described for Target Variable 1, detailed results and intermediate tables are omitted here to avoid redundancy. However, the same evaluation metrics (AUC, Accuracy, Log-loss, Precision, Recall, and F1-score) were used, and AdaBoost was again selected as the most suitable model after optimization.

Deployment of the Model

A crucial step in this project was to outline how the model could be deployed in a real-world environment. Deployment ensures that the model is not only evaluated academically but also used to support daily decision-making in the restaurant. The process can be divided into three main stages:

1. **Data Input and Preparation**
    - Import raw data files such as `Restaurant-2-orders`, `Restaurant-2-products-price`, `Weatherstats_london_daily`, and `calendar`.
    - Store the data in a structured SQL flat file for efficient retrieval and integration.
2. **Data Cleaning and Quality Assurance (QA)**
    - Open the dedicated Python notebook (`EDA_FinalProject_TargetValue1\2\3`) to clean the data and remove missing values.
    - Ensure data quality by checking for errors and validating that no inconsistent records remain before passing data into the model.
3. **Model Training, Evaluation, and Deployment**
    - Use the prepared machine learning notebooks (`ML_FinalProject_TargetValue1\2\3`) to retrain the AdaBoost model and evaluate its performance.
    - Save the best-performing model (measured by AUC and validation metrics) for deployment.
    - Integrate the model into the restaurant's operational system: the chef or supply manager would receive daily or weekly forecasts through a simple dashboard or mobile app.
    - These forecasts would indicate which products are expected to be in demand over the next three days, ensuring accurate inventory planning, fresh ingredients, and reduced waste.

To maintain reliability, the model should be retrained periodically whenever new data is collected, or if its predictive performance declines. This would allow the restaurant to benefit from a continuously updated, trustworthy decision-support system.

Conclusion

This project set out to address a practical yet highly relevant problem: predicting the demand for products in an Indian restaurant in London over a three-day horizon. The challenge lay not only in selecting an appropriate prediction target, but also in collecting, cleaning, and integrating heterogeneous data sources including orders, product prices, weather conditions, and calendar events into a structured SQL flat file. The data preparation and exploratory analysis (EDA) phase was among the most demanding, ensuring that the input fed into the models was both reliable and representative.

Several classification models were trained and compared, with careful attention to avoiding overfitting and ensuring generalization. Through systematic experimentation, AdaBoost emerged as the most suitable model. Further fine-tuning of its hyper parameters (such as `n_estimators`, `learning_rate`, and base estimator depth) significantly improved its performance, with validation AUC gains of more than 5% compared to the baseline. Importantly, the model maintained consistent results on the test set, demonstrating robustness and stability.

Beyond the technical performance, the study highlights the potential real-world impact of deploying such a model. By providing accurate three-day demand forecasts, the restaurant can optimize its supply chain, reduce waste, and ensure product availability, directly supporting operational efficiency and customer satisfaction.

In conclusion, this project illustrates how data-driven approaches, when coupled with rigorous preprocessing and model selection, can deliver tangible value in the food industry. Future extensions could incorporate additional external factors (e.g., economic indicators, local events, or unexpected disruptions) to further improve forecasting accuracy and adaptability.