SENTIMENT ANALYSIS FOR JUMIA REVIEWS & SMARTPHONE PRICE PREDICTION SYSTEM

Machine Learning Project Final Writeup

Aymane Maghouti. (aymane.maghouti@etu.uae.ac.ma)

CNE: S131419078

Abstract:

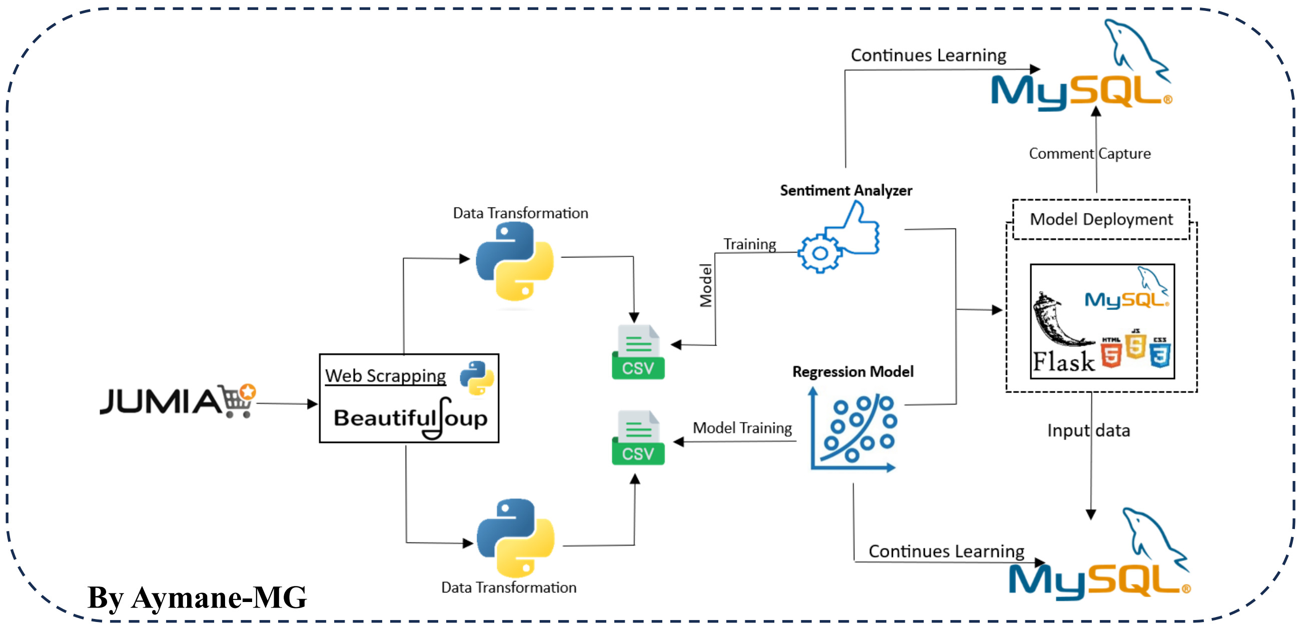
This project aims to elevate the user experience on the Jumia platform by implementing two machine learning systems. The first system focuses on sentiment analysis of Jumia reviews, involving data collection, transformation, and cleaning. Various models, including Multinomial Naive Bayes, SVM, and Logistic Regression, are developed and evaluated using metrics such as accuracy and confusion matrix. The optimal model is integrated into a web application, enabling real-time verification of product recommendations by analyzing associated reviews and gathering user votes. The primary objective is to deliver a seamless and informed shopping experience on Jumia through the application of machine learning techniques and user-friendly web interfaces. In the second system, the project centers on predicting smartphone prices by collecting and processing data from the Jumia website. Algorithms such as RandomForestRegression, XGBregression, linear regression, ridge regression, and lasso regression are employed. Model performance is assessed using metrics like MSE, MAE, RMSE, and R-square. The optimal model is then integrated into a user-friendly web application using Flask, JavaScript, and HTML/CSS.

1.Introduction:

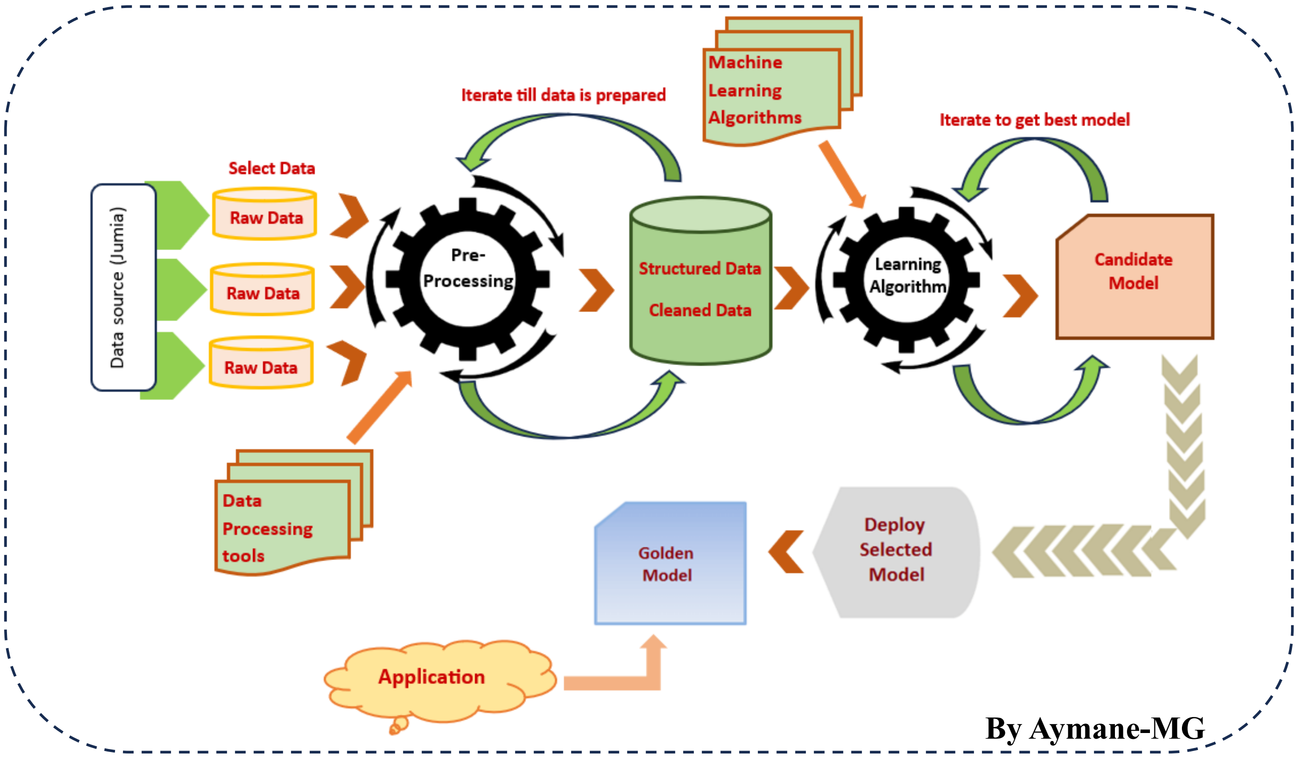
In the realm of e-commerce, user experience plays a pivotal role, and two key challenges faced by shoppers on platforms like Jumia are unpredictable smartphone pricing and the overwhelming volume of product reviews. The fluctuating nature of smartphone prices, influenced by factors like brand and specifications, can impede users from making informed purchase decisions. Additionally, navigating through numerous product reviews to discern sentiment can be daunting. Recognizing these challenges, our project employs machine learning techniques to develop systems for predicting smartphone prices and performing sentiment analysis on Jumia reviews.

The unpredictable pricing of smartphones on Jumia is affected by various factors, creating uncertainty for potential buyers. Similarly, the abundance of product reviews poses a challenge for users seeking to understand the sentiment associated with a product. To address this, our project utilizes machine learning algorithms, including RandomForestRegression, XGBregression, and sentiment analysis models such as Multinomial Naive Bayes and Logistic Regression. By doing so, we aim to provide users with reliable price predictions and insightful sentiment analysis, ultimately simplifying the decision-making process. In comparison to related articles, this project aligns with the sentiment analysis aspect discussed in [3], where sentiment analysis is applied to assess recommendations in the context of e-learning. Similarly, our project employs sentiment analysis for Jumia reviews to enhance product recommendations. Additionally, the utilization of machine learning in both systems draws parallels with the role of sentiment analysis and opinion classification in recommendation systems outlined in [2]. The project contributes to the evolution of recommendation systems by providing real-time verification and personalized shopping experiences, merging sentiment analysis with e-commerce dynamics.

The Architecture



Model building:

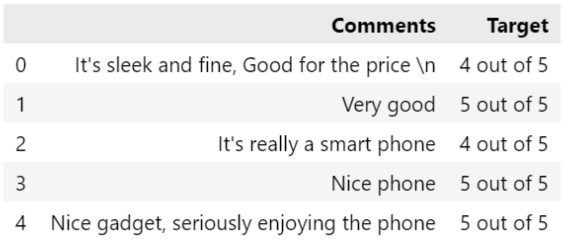


2. System 1: Sentiment Analysis for Jumia Reviews

2.1 Data Collection:

For the Sentiment Analysis model targeting Jumia reviews, the initial step involves data collection from the Jumia website. This process is facilitated using Beautiful Soup, a web scraping tool, enabling the extraction of relevant textual data from product reviews. The collected data encompasses user-generated reviews containing sentiments that range from positive to negative, providing a diverse and representative dataset for training and evaluating the machine learning model.

Example of dataset:



	Comments	Target
0	It's sleek and fine, Good for the price \n	4 out of 5
1	Very good	5 out of 5
2	It's really a smart phone	4 out of 5
3	Nice phone	5 out of 5
4	Nice gadget, seriously enjoying the phone	5 out of 5

2.2 Data Transformation and Cleaning

Following data collection, the next crucial step is data transformation and cleaning to prepare the dataset for sentiment analysis. The objective is to standardize the format, eliminate noise, and create a well-structured dataset for model training. The following transformations are applied:

- Encoding Sentiments:

* Reviews with ratings 1 and 2 are encoded as -1 representing negative sentiments.

* Reviews with ratings 4 and 5 are encoded as 1, representing positive sentiments.

* Reviews with a rating of 3 are excluded from the dataset as they are considered neutral.

- Text Preprocessing:

- Lowercasing: Convert all text to lowercase to ensure consistency in the dataset.

- Removing Punctuation and Numbers: Extraneous characters, including punctuation and numbers, are removed to focus on the textual content.

- Stop Words Removal: Common stop words are removed to reduce noise and improve the model's focus on meaningful words.

- Lemmatization: Words are lemmatized to reduce inflections and variations to their base or root form.

- Handling Rare Words: Rare words that may not contribute significantly to sentiment analysis are identified and removed to streamline the dataset.

About the dataset:

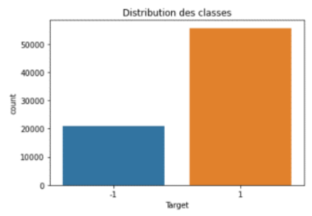


Fig.1: The distribution of classes

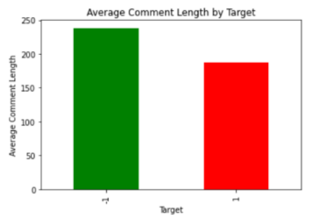


Fig.2: Average comment length by target

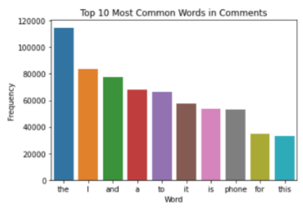


Fig.3: Before cleaning

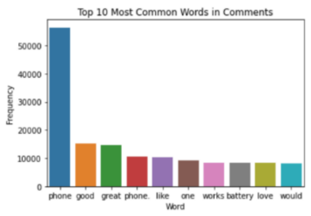


Fig.4: After cleaning

2.3 Comments Classification Approach:

After data transformation and cleaning, the next step involves preparing the data for model training. The process includes converting the textual data into numerical representations and employing the train-test split method to assess model performance. Here's the breakdown of the methodology:

TF-IDF Vectorization:

Text data is converted into numerical features using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer. This technique assigns weights to words based on their importance in the corpus of reviews, capturing the significance of each word in relation to the entire dataset. The TF-IDF vectorizer is employed with the parameter ngram_range=(1,3), capturing unigrams, bigrams, and trigrams. This allows the model to consider not only individual words but also combinations of adjacent words, providing a more comprehensive representation of the textual context.

Term Frequency measures how often a term (word) appears in a document. It is calculated as the ratio of the number of times a term appears in a document to the total number of terms in the document

$$\textbf{Term Frequency (TF):}[\mathrm{TF}(t,d) = \frac{\text{Number of occurrences of term } t \text{ in document } d}{\text{Total number of terms in document } d}]$$

Inverse Document Frequency measures the rarity of a term across the entire document corpus. It is calculated as the logarithm of the ratio of the total number of documents in the corpus to the number of documents that contain the term t.

$$\textbf{Inverse Document Frequency (IDF):}[\mathrm{IDF}(t,D) = \log\left(\frac{\text{Total number of documents in the corpus } |D|}{\text{Number of documents containing term } t+1}\right)]$$

The TF-IDF weight is higher for terms that are frequent in the current document but rare across the entire corpus, indicating their importance in describing the content of that document.

$$\textbf{TF-IDF Weight:}[\text{TF-IDF}(t,d,D) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t,D)]$$

Train-Test Split:

The dataset is split into training and testing sets using the train-test split method. The parameter (stratify=data['Target']) ensures that the distribution of positive and negative sentiments is maintained in both the training and testing sets, preserving the balance of sentiments for robust model evaluation.

2.4 Algorithms:

Multinomial Naive Bayes:

Naive Bayes is one of the most common generative learning algorithms for classification problems. This algorithm assumes that x_i s are conditionally independent given y, which is called Naive Bayes assumption.

$$\textbf{Likelihood:}[P(X \mid Y=c) = \prod_{j=1}^{n} P(X_j \mid Y=c)^{x_{ij}}]$$

We also incorporated Laplace Smoothing in our model to make it work better. The prediction of an example is given by the formula below:

$$\textbf{Decision Rule:}[\text{Prediction} = \arg\max_{c} P(Y=c \mid X)]$$

Support Vector Machine (SVM):

In the case of classification, SVM aims to find the hyperplane that best separates positive and negative sentiment data points, maximizing the margin between the two classes.

Decision Function:
The decision function for SVM is defined as the dot product of the feature vector x and the weight vector w, plus a bias term b:

$$\textbf{Decision Function:}[f(x) = \langle w, x \rangle + b]$$

Where:
- f(x) is the decision function.
- w is the weight vector
- b is the bias term.
- x is the input feature vector.

SVM aims to maximize the margin between the support vectors, which are the data points closest to the decision boundary.

$$\textbf{Margin:}[\text{Margin} = \frac{2}{||w||}]$$

Where:

• ||w|| is the Euclidean norm of the weight vector.

<u>Logistic Regression:</u>

Logistic regression is a simple algorithm that is commonly used in binary classification. Due to its efficiency, it is the first model we selected to do the classification. The hypothesis of Logistic Regression algorithm is:

$$\textbf{Logistic Regression Hypothesis:}[h_\theta(x) = \frac{1}{1 + e^{-(\theta_0+\theta_1 x_1+\theta_2 x_2+\ldots+\theta_n x_n)}}]$$

The algorithm optimizes θ by maximizing the following log likelihood function:

$$\text{Log-Likelihood}(\theta) = \sum_{i=1}^{m} [y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

2.5 Selected Model:

In a confusion matrix, we describe the performance of a classification model. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (vice versa). There are four basic terms in a confusion matrix:
- TP (true positive): an outcome where the model correctly predicts the positive class.
- TN (true negative): an outcome where the model correctly predicts the negative class.
- FP (false positive): an outcome where the model incorrectly predicts the positive class.
- FN (false negative): an outcome where the model incorrectly predicts the negative class.

Here we select two metrics: accuracy and F1 score

<u>Accuracy:</u> The proportion we have predicted right.

$$\text{Accuracy} = \frac{TP + TN}{total}$$

<u>F1-Score:</u> the harmonic mean of precision and recall, providing a balanced measure.

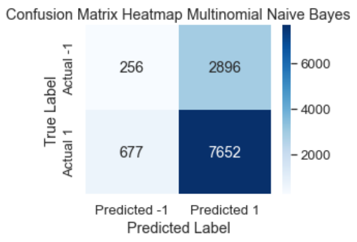
$$\text{Precision} = \frac{TP}{TP + FP}$$

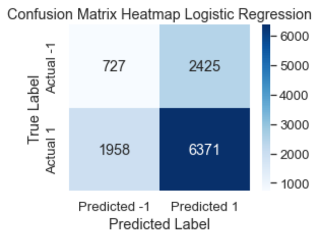
$$\text{Recall} = \frac{TP}{TP + FN}$$

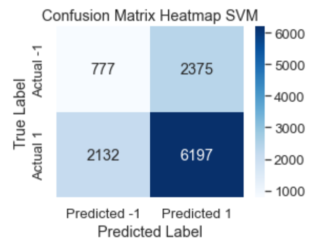
$$\text{F1} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Dataset	Training Set (70%)	Validation Set (15%)	Test Set (15%)
76539	53577	11481	11481

Confusion Matrix:







Metric / Model	Accuracy	Precision	Recall	F1-score
SVM	**0.9470**	**0.9537**	0.9747	**0.9641**
Multinomial Naive Bayes	0.81	0.7942	**0.9988**	0.8848
Logistic Regression	0.9304	0.9313	0.9767	0.9535

Analysis and discussion:

- SVM has the highest overall accuracy at 94.70%, indicating that 94.70% of the model's predictions are correct.

- SVM has the highest precision at 95.37%, indicating the proportion of true positives among all positive predictions.

- Logistic Regression also has a high precision at 93.13%.
- Multinomial Naive Bayes has exceptionally high recall at 99.88%, suggesting it identifies almost all true positive examples. SVM also has a high recall at 97.47%.
- SVM has the highest F1-score at 96.41%.
- SVM appears to be the best overall model, offering a balance between accuracy, precision, recall, and F1-score. Multinomial Naive Bayes shows high precision and recall but has a lower F1-score compared to SVM, indicating a potential imbalance between precision and recall. Logistic Regression performs well but is slightly inferior to SVM in terms of accuracy, precision, recall, and F1-score.

Hyperparameter Tuning

Support Vector Machine

RandomizedSearchCV is a hyperparameter tuning technique in machine learning that, instead of exhaustively searching through all possible hyperparameter combinations like GridSearchCV, randomly samples a specified number of combinations from predefined hyperparameter distributions. This method is particularly useful when the search space is large, as it provides a more computationally efficient approach to finding optimal hyperparameters for a model. So, in our case we will use the two hyperparameters:

- 'C': This is the regularization strength, and it's the inverse of the regularization parameter. Smaller values specify stronger regularization. The represent different magnitudes of the regularization strength, allowing the grid search to find the optimal value. (C=1/ λ)

- 'Kernel': The kernel function determines the type of decision boundary that the SVM will use. It transforms the input features into a higher-dimensional space, allowing the algorithm to find a nonlinear decision boundary in the original feature space.

Radial Basis Function (RBF) Kernel:

$$K(x, y) = \exp\left(-\gamma \cdot \|x - y\|^2\right)$$

Polynomial Kernel:

$$K(x, y) = (\alpha \cdot x^T \cdot y + \beta)^d$$

Where:
K(x,y) represents the kernel function's output for two input vectors x and y.

Hyperparameter	different values
C	0.1, 1, 2, 10
kernel	'rbf', 'poly'

Results :

Hyperparameter	Value
C	2
kernel	'rbf'

Final evaluation on the test set:

After choosing the best model we find the following result on the test set:

```
Classification Report:
              precision    recall  f1-score

          -1       0.92      0.85      0.88
           1       0.94      0.97      0.96
```

* Precision = 0.93
* recall = 0.91
* F1-score = 0.92

Logistic Regression

we will use the two hyperparameters:

- **C**: This is the regularization strength, and it's the inverse of the regularization parameter. (C=1/ λ)

Penalty:
L1 Regularization (penalty='l1'): In L1 regularization, the penalty term is the absolute value of the coefficients (L1 norm).

L2 Regularization (penalty='l2'): In L2 regularization, the penalty term is the squared value of the coefficients (L2 norm).

Hyperparameter	different values
C	0.001, 0.01, 0.1, 1, 10, 100
Penalty	'l1', 'l2'

Results:

Hyperparameter	Value
C	10
kernel	'l2'

Final evaluation on the test set:

After choosing the best model we find the following result on the test set:

```
Final Accuracy on Test Set: 0.9389
Classification Report:
              precision    recall  f1-score

          -1       0.92      0.85      0.88
           1       0.95      0.97      0.96
```

* Precision = 0.935
* recall = 0.91
* F1-score = 0.92

Multinomial Naive Bayes:

we will use the two hyperparameters:

alpha parameter: is a smoothing parameter that is used to handle the issue of zero probabilities in the likelihood estimation.

- When `alpha` is zero, no smoothing is applied, and you rely solely on the observed frequencies in the training data. However, this can lead to zero probabilities for unseen features.

- Setting a small positive value for `alpha` (commonly 1) helps to smooth the probability estimates and avoids assigning zero probability to unseen features.

fit_prior parameter: is a boolean parameter that determines whether to learn class prior probabilities from the training data (fit_prior=True) or to use uniform priors (fit_prior=False).

- If `fit_prior=True`, the model estimates class prior probabilities based on the frequency of each class in the training data.

- If `fit_prior=False`, uniform prior probabilities are used, assuming that each class is equally likely.

Hyperparameter	different values
alpha	0.001, 0.01, 0.1, 1, 10, 100
fit_priori	'True', 'False'

Results:

Hyperparameter	Value
alpha	0.1
fit_priori	'True'

Final evaluation on the test set:

After choosing the best model we find the following result on the test set:

```
Final Accuracy on Test Set: 0.9216
Classification Report:
              precision    recall  f1-score

          -1       0.96      0.74      0.84
           1       0.91      0.99      0.95
```

* Precision = 0.92
* recall = 0.935
* F1-score = 0.895

→ SVM still the best model for our problem.

2.6 Model Deployment in Web Environments

We created a web user interface, here is the architecture how it works:

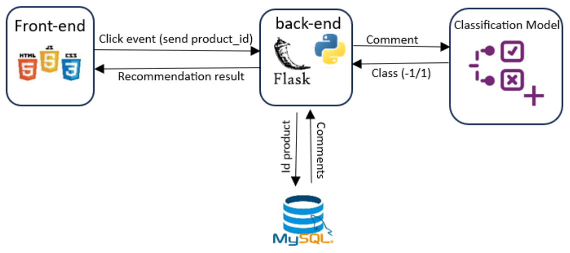


Fig.5: web app architecture for product recommendation

The recommendation result is based on comments classification. For example, if the positive comments outnumber the negative comments, the product is recommended for purchase; otherwise, it is not. However, the application will not provide a result in two cases: when the positive comments are equal to the negative comments or when the product has no comments.

Additionally, the web application provides users with a space to give their comments about a product. The application works in real-time, meaning the recommendation result could be changed dynamically based on the incoming feedback.

2.7 Limitations and Future Work:

The project is subject to certain limitations that warrant consideration. Firstly, in scenarios with a substantial volume of comments, the recommendation process may experience delays, affecting the responsiveness of the system. Additionally, the system's performance may be compromised when faced with grammatically incorrect comments, potentially leading to inaccurate recommendations. To address these limitations, future work could explore alternative techniques and algorithms in natural language processing (NLP) to enhance comment understanding, improving the system's robustness in the face of varied linguistic styles. Furthermore, incorporating frameworks like SparkML could offer faster processing capabilities, ensuring efficient recommendation generation even in the presence of a large number of comments. These advancements would contribute to a more responsive and accurate recommendation system, overcoming current constraints and providing a more seamless user experience.

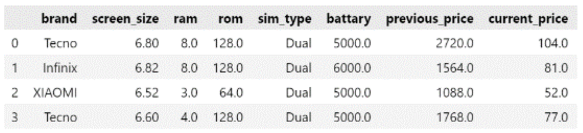
3. System 2: Smartphone Price Prediction System

3.1 Data Collection and Cleaning:

The primary source of data for this machine learning project was Jumia, a prominent online e-commerce platform specializing in consumer electronics, including smartphones. The data collection focused specifically on the "Mobile Phones" category, where detailed product information was gathered for analysis and subsequent price prediction.
Web scraping techniques were employed to extract relevant information from product. after cleaning the data, we got these attributes:

- Brand
- RAM
- Screen Size
- ROM
- Sim type
- Battery
- current Price
- previous price

Example of the dataset:



	brand	screen_size	ram	rom	sim_type	battary	previous_price	current_price
0	Tecno	6.80	8.0	128.0	Dual	5000.0	2720.0	104.0
1	Infinix	6.82	8.0	128.0	Dual	6000.0	1564.0	81.0
2	XIAOMI	6.52	3.0	64.0	Dual	5000.0	1088.0	52.0
3	Tecno	6.60	4.0	128.0	Dual	5000.0	1768.0	77.0

3.2 Data Transformation:

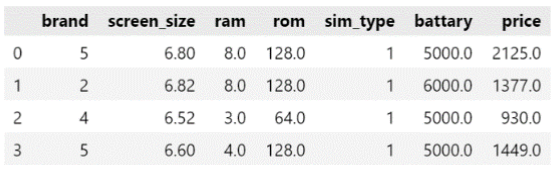
Encoding Categorical Variables

To incorporate categorical variables into the machine learning model, the "brand" and "sim_type" features were encoded. The encoding process was essential as machine learning algorithms typically require numerical input.

Adding a Mean Price Column

A new feature, "price," was created as the mean of the "current_price" and "previous_price" columns. This aggregated feature aims to capture a representative pricing value for each smartphone.

Example of the final Transformed Dataset:



	brand	screen_size	ram	rom	sim_type	battary	price
0	5	6.80	8.0	128.0	1	5000.0	2125.0
1	2	6.82	8.0	128.0	1	6000.0	1377.0
2	4	6.52	3.0	64.0	1	5000.0	930.0
3	5	6.60	4.0	128.0	1	5000.0	1449.0

About the dataset:

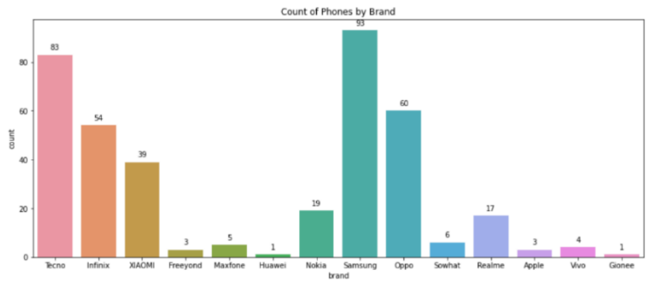


Fig.6: Count of phones by brand

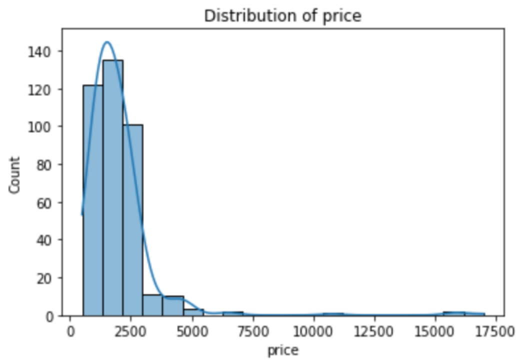


Fig.7: Distribution of price

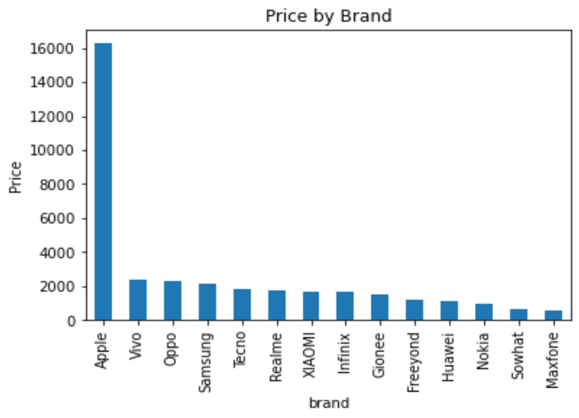
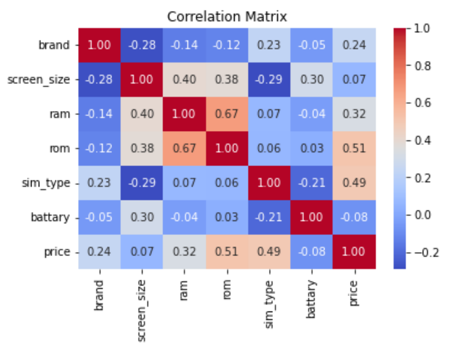


Fig.8: Price By Brand



	brand	screen_size	ram	rom	sim_type	battary	price
brand	1.00	-0.28	-0.14	-0.12	0.23	-0.05	0.24
screen_size	-0.28	1.00	0.40	0.38	-0.29	0.30	0.07
ram	-0.14	0.40	1.00	0.67	0.07	-0.04	0.32
rom	-0.12	0.38	0.67	1.00	0.06	0.03	0.51
sim_type	0.23	-0.29	0.07	0.06	1.00	-0.21	0.49
battary	-0.05	0.30	-0.04	0.03	-0.21	1.00	-0.08
price	0.24	0.07	0.32	0.51	0.49	-0.08	1.00

Fig.9: Correlation Matrix

3.3 Price Prediction Approach:

In order to robustly assess the performance of regression models and ensure their generalizability, a systematic approach was adopted that involved the use of cross-validation coupled with the implementation of diverse regression algorithms. The dataset, consisting of 388 rows capturing various smartphone features, was subjected to a meticulous cross-validation procedure. Specifically, a 5-fold cross-validation scheme was employed, where the dataset was partitioned into five subsets, allowing each fold to serve alternately as a training set and a validation set. This methodological choice was motivated by the relatively moderate size of the dataset, aiming to strike a balance between computational efficiency and obtaining reliable estimates of model performance.

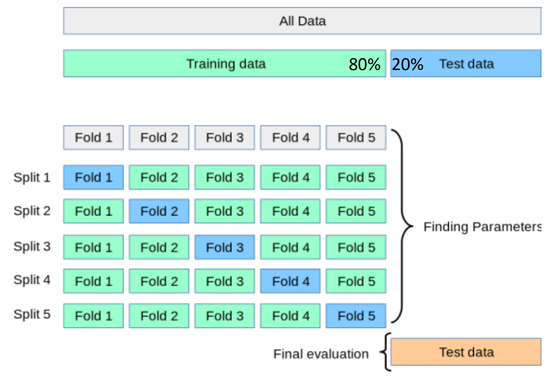


Fig.10: Cross-Validation strategy

3.4 Algorithms:

Random forest regression

Random Forest is an ensemble learning method that operates by constructing a multitude of decision trees at training time and outputting the average prediction of the individual trees for regression tasks.

The prediction of the random forest is an average (or weighted average) of predictions from individual decision trees. Let $\hat{y}_i$ be the prediction from the i-th tree, and N be the number of trees.

$$\hat{y}_{\mathrm{RF}}(x) = \frac{1}{N}\sum_{i=1}^{N} \hat{y}_i(x)$$

Linear Regression (with Lasso & Ridge extensions)

Linear Regression models the relationship between the dependent variable and one or more independent variables using a linear equation. Lasso and Ridge are regularization techniques applied to linear regression.

$$\hat{y}_{\mathrm{linear}}(x) = \lambda_0 + \lambda_1 \cdot x_1 + \lambda_2 \cdot x_2 + \ldots + \lambda_p \cdot x_p$$

- Lasso adds a regularization term to the linear regression objective function (l1):

$$\hat{y}_{\mathrm{lasso}}(x) = \lambda_0 + \lambda_1 \cdot x_1 + \lambda_2 \cdot x_2 + \ldots + \lambda_p \cdot x_p + \lambda \sum_{i=1}^{p} |\theta_i|$$

- Ridge also adds a regularization term but uses the squared magnitude of coefficients(l2):

$$\hat{y}_{\mathrm{ridge}}(x) = \lambda_0 + \lambda_1 \cdot x_1 + \lambda_2 \cdot x_2 + \ldots + \lambda_p \cdot x_p + \lambda \sum_{i=1}^{p} \theta_i^2$$

XGB regressor

XGBoost uses a boosting approach where each subsequent model corrects errors made by the previous ones. The final prediction is a sum of contributions from individual trees:

$$\hat{y}_{\mathrm{XGBoost}}(x) = \sum_{i=1}^{N} f_i(x)$$

where N is the number of boosting rounds, and $f_i(x)$ is the prediction from the i-th tree.

3.5 Selected Model:

Mean absolute error (MAE), mean squared error (MSE) and R2 score were used to evaluate the trained models.

1. Mean Squared Error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where n is the number of observations, y_i is the actual value, and $\hat{y}_i$ is the predicted value.

2. Mean Absolute Error (MAE):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

where n is the number of observations, y_i is the actual value, and $\hat{y}_i$ is the predicted value.

3. R-squared (R^2):

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where n is the number of observations, y_i is the actual value, and $\hat{y}_i$ is the predicted value, and $\bar{y}$ is the mean of the actual values.

Training (310 examples) and validation (62 examples) splits were used to choose the best-performing models within each category. The test set, containing 78 examples, was used to provide an unbiased estimate of error, with the final models trained on both train and validation splits. Results for the final models are provided below:

Model	Train split			Test split		
	MAE	MSE	R^2 Score	MAE	MSE	R^2 Score
Linear Reg.	455.99	788524.94	0.49	718.37	3581104.00	0.45
Ridge Reg.	454.28	788628.77	0.49	719.89	3595019.08	0.44
Lasso Reg.	448.18	791562.43	0.48	729.61	3700416.16	0.43
Random Forest Reg.	147.62	189322.16	0.87	415.59	1462603.35	0.77
XGB Reg.	**91.62**	**21701.36**	**0.98**	**342.08**	**859561.33**	**0.86**

Analysis and discussion:

- Linear regression models (Linear, Ridge, Lasso) show relatively similar performances, suggesting they may not capture the complexity of the problem optimally.

- Ensemble models like Random Forest and XGBoost demonstrate significantly superior ability to capture non-linear relationships in the data.

- XGBoost appears to be the best model in terms of MAE, MSE, and R^2, indicating it provides the best prediction on the test set.

Based on the presented results, the XGBoost model seems to be the optimal choice for predicting phone prices in this particular context. It outperforms other models on both training and test sets, showcasing exceptional ability to generalize to new data.

Hyperparameter Tuning

Grid search is a hyperparameter tuning method used to find the optimal hyperparameter values for a machine learning model.

XGB regressor:

we will use the three hyperparameters:

n_estimators:

- Represents the number of trees (weak learners) to be built in the ensemble.
- Increasing `n_estimators` generally improves the model's performance, but it also increases computation time.

learning_rate:

- Controls the contribution of each tree to the final prediction.
- A lower learning rate requires more trees to achieve the same level of model complexity.
- It's a regularization technique, and smaller values often lead to better generalization.

max_depth:

- Specifies the maximum depth of each tree in the ensemble.
- A deeper tree can capture more complex patterns but may lead to overfitting.
- It's important to tune `max_depth` to find the right balance between model complexity and generalization.

Hyperparameter	different values
n_estimators	100, 200, 300
learning_rate	0.01, 0.1, 0.2
max_depth	3, 4, 5

Results

Hyperparameter	different values
n_estimators	200
learning_rate	0.2
max_depth	4

Final evaluation on the test set:

We find the following result on the test set:

```
Test Set Metrics:
Mean Absolute Error (MAE): 367.6569049541767
Mean Squared Error (MSE): 995017.7511480671
R-squared (R²): 0.8475743768175301
```

Linear Regression – Lasso - Ridge:

we will use the one hyperparameter:

alpha: is the regularization parameter, controlling the strength of the regularization. A higher alpha lead to more regularization.

Hyperparameter	different values
alpha	logspace(-4, 4, 9)

(logspace(-4,4,9) generates 9 values from 10^{-4} to 10^4 [0.0001, 0.001, 0.01, 1, 10, 100, 1000, 10000])

Results
Lasso :

Hyperparameter	different values
alpha	0.0001

we find the following result on the test set:

```
Test Set Metrics:
Mean Absolute Error (MAE): 718.3708537389034
Mean Squared Error (MSE): 3581105.38416139
R-squared (R²): 0.4514145911134899
```

Ridge

Hyperparameter	different values
alpha	10.0

we find the following result on the test set:

```
Test Set Metrics:
Mean Absolute Error (MAE): 778.3742871222676
Mean Squared Error (MSE): 4175739.408022133
R-squared (R²): 0.3603232899302753
```

Random forest regression

we will use the three hyperparameters:

n_estimators: Number of trees in the forest.

max_depth: Maximum depth of the individual trees.

min_samples_split: The minimum number of samples required to split an internal node.

Hyperparameter	different values
n_estimators	50,100,200
max_depth	10,20,30
min_samples_split	2,5,10

Results

Hyperparameter	different values
n_estimators	50
max_depth	30
min_samples_split	5

we find the following result on the test set:

```
Test Set Metrics:
Mean Absolute Error (MAE): 447.41021408188095
Mean Squared Error (MSE): 1768038.4123612617
R-squared (R²): 0.7291562321337854
```

→Still the XGBoost model is the best model for our problem.

3.6 Model Deployment in Web Environments

We created a web user interface, here is the architecture how it works:

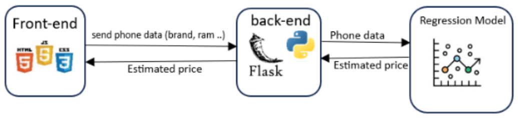


Fig.11: web app architecture for price estimation

The web app is designed with a user-friendly interface, guiding the user through the input process to ensure accurate and meaningful predictions. Users can easily enter the specifications of the phone they are interested in, and the application seamlessly communicates with the trained XGBoost regression model to generate a price estimate.

3.7 Limitations and Future Work:

While the developed phone price estimation system has demonstrated efficacy, certain limitations warrant consideration. The model's performance is contingent on the quality and diversity of the training dataset; thus, future efforts should focus on acquiring a more extensive and representative dataset. Feature enrichment by incorporating additional attributes, such as customer reviews and regional market trends, could enhance the model's predictive capabilities. Moreover, exploring advanced machine learning models, along with sophisticated hyperparameter tuning techniques, offers potential for improved accuracy. Addressing these limitations, future work involves continuous refinement of the model through data expansion, feature augmentation, and the exploration of cutting-edge methodologies, ensuring the system's adaptability to evolving market dynamics.

4.Conclusion:

In conclusion, our ventures into sentiment analysis for Jumia reviews and phone price estimation represent significant milestones in harnessing machine learning for enhanced user experiences. The sentiment analysis system, integrating Beautiful Soup, TF-IDF vectorization, and various classifiers, has demonstrated effective sentiment classification. Similarly, the phone price estimation system, powered by an XGBoost regression model, has provided users with accessible and accurate pricing insights, albeit with room for improvement in data completeness and feature enrichment. Both systems underscore the ongoing commitment to advancing machine learning applications, with future endeavors focusing on overcoming limitations and ensuring continued relevance in their respective domains.

5.Github Repository

Welcome to check out the code here: [link]

6.References:

[1] Indriani Sitorus, Analytics Vidhya, Medium, [link], Aug 27, 2020.

[2] Ziani-Amel, [link], January 26,2020.

[3] Lazid Lydia, Hacid Kahina, Université Mouloud Mammeri, [link], 2013.

[4] Kasun Dissanayake, Towards Dev, Medium , [link], Dec 14, 2023.