**METHODOLOGY**

The system model is divided into four main phases: data collection, training, testing, and evaluation. We collected necessary data, preprocessed the data so that they would be usable with our chosen machine learning algorithms and subsequently trained and tested the model so that we could use the model for future classification of processors.

**Data Collection**

**Visualization**

**Model Evaluation using the testing dataset to assess their classification performance.**

**Model Selection and Training**

**Data Exploration**

**Data Integration and Feature Engineering**

**Testing and Optimization**

**Figure 1.0:** Flow of operations in the created system

This methodology outlines the steps involved in building a machine learning model that can distinguish between AMD and Intel processors, using publicly available specification data. Here's a detailed breakdown of each step:

1. **Data Collection:**

The first step involves gathering data. We need two separate datasets, one containing specifications for AMD processors and another for Intel processors, typically in CSV format. Once we have the data, we ensure consistency by standardizing the column names between the two datasets. Then, we select features most relevant to processor classification, such as product name, release date, core/thread count, clock speed, cache size, and thermal design power (TDP).

Next, we address missing data. In the Intel dataset, where data is assumed to be mostly complete, we might choose to simply remove rows with missing values. However, for the AMD dataset, we can impute missing values by replacing them with the average value for each numeric feature. Finally, we create a new variable to represent the processor brand. We assign a value of "0" to AMD processors and "1" to Intel processors.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | launch Date | numCores | numThreads | graphicsModel | baseClock | | TDP | Cache |
| AMD Ryzen™ 5 6600H | 2022 | 6 | 12 | AMD Radeon™ 660M | 3.3 | 45 | | 384 |
| AMD Ryzen™ 5 6600HS​ | 2022 | 6 | 12 | AMD Radeon™ 660M | 3.3 | | 35 | 384 |
| AMD Ryzen™ 5 6600U | 2022 | 6 | 12 | AMD Radeon™ 660M | 2.9 | | 15 | 384 |
| AMD Ryzen™ 5 PRO 6650H | 2022 | 6 | 12 | AMD Radeon™ 660M | 3.3 | | 45 | 384 |
| Atom x6200FE | 2021 | 2 |  | Intel UHD Graphics for 10th Generation Intel Core Processors | 1 | |  | 1.5 |
| Atom x6211E | 2021 | 2 |  | Intel UHD Graphics for 10th Generation Intel Core Processors | 1.3 | |  | 1.5 |
| Atom x6212RE | 2021 | 2 |  | Intel UHD Graphics for 11th Generation Intel Core Processors | 1.2 | |  | 1.5 |
| Atom x6413E | 2021 | 4 |  | Intel UHD Graphics for 10th Generation Intel Core Processors | 1.5 | |  | 1.5 |
| Atom x6414RE | 2021 | 4 |  | Intel UHD Graphics for 10th Generation Intel Core Processors | 1.5 | |  | 1.5 |
| Atom x6425E | 2021 | 4 |  | Intel UHD Graphics | 2 | |  | 1.5 |

**Table 1.0**: Sample of data collected

1. **Data Exploration:**

After cleaning the data, we performed Exploratory Data Analysis (EDA) to understand the data distribution and relationships between features. This helps us gain insights into the data and identify any potential issues. Common EDA techniques include:

* Visualizing the distribution of the target variable (processor brand) using count plots to see how many processors belong to each category (AMD or Intel).
* Creating pair plots to visually explore the relationships between different features, like core count and clock speed.
* Calculating correlation coefficients, such as Spearman's correlation, to quantify the linear relationships between features. This helps us understand how features might influence each other.

1. **Data Integration and Feature Engineering:**

Once we have a good understanding of the data, we combine the preprocessed AMD and Intel dataframes into a single, unified dataframe. This allows us to train a single model on all the data.

However, before training the model, we need to perform some feature engineering. Categorical data, like product names, can't be directly used by machine learning models. We addressed this by using a technique called Label Encoding. This process assigns a unique numerical label to each category (e.g., "A10-5700" gets label 1, "Core i7" gets label 2).

Additionally, for numerical features like core count and clock speed, we use a technique called Standardization. This ensures that all features have a similar scale (typically zero mean and unit standard deviation). This is important because some models can be sensitive to the scale of features, and standardization helps prevent features with larger scales from dominating the model during training.

1. **Model Training and Evaluation:**

Now that our data is prepared, we can move on to training the model. Here's how we went about it:

**Train-Test Split:** We split the merged dataframe into two sets - a training set and a testing set. Typically, a 70/30 split is used, where 70% of the data is used for training the model and the remaining 30% is used for testing its performance on unseen data.

**Model Selection and Training:** We choose a suitable machine learning model for binary classification, which means it can predict one of two classes (AMD or Intel in this case). In this system, we used a Support Vector Machine (SVM) model. The model is then trained on the training set, allowing it to learn the patterns and relationships within the data.

**Model Evaluation:** Once trained, we evaluate the model's performance on the unseen testing set. This helps us assess how well the model generalizes to new data. We use various metrics to evaluate the model, including:

* Accuracy: This metric tells us the proportion of correctly classified instances (correctly predicted AMD or Intel processors).

95.41% - This indicates that the model correctly classified over 95% of the processors in the testing set.

* F1-Score: This metric provides a balance between precision (proportion of true positives among predicted positives) and recall (proportion of true positives identified).

F1 = 2 \*

0.963 - This is a harmonic mean of precision and recall, balancing between how well the model identifies true positives (correctly predicts AMD/Intel) and avoids false positives (incorrectly predicts AMD/Intel). A score close to 1 signifies a well-balanced model.

* ROC-AUC Score: This score represents the Area Under the Curve of the Receiver Operating Characteristic (ROC) curve. It helps us evaluate the model's ability to distinguish between AMD and Intel processors across all classification thresholds.

AUC=∫10TPR(t)dFPR(t)

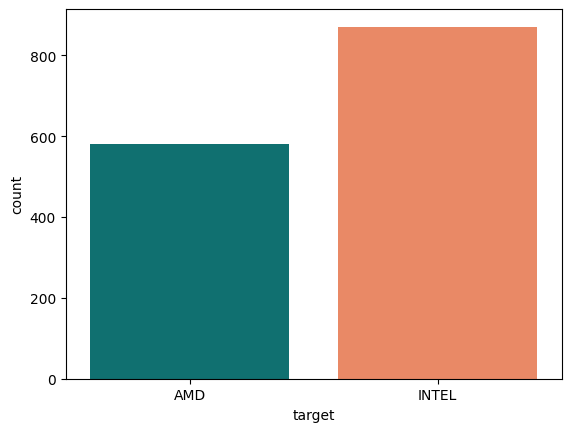
0.943 - Area under the Curve of the Receiver Operating Characteristic (ROC) curve. This score tells us how well the model can distinguish between AMD and Intel processors across all possible classification thresholds. A score closer to 1 indicates better discrimination between the two classes.

**Visualization and Reporting:** Finally, we visualize the model's performance using:

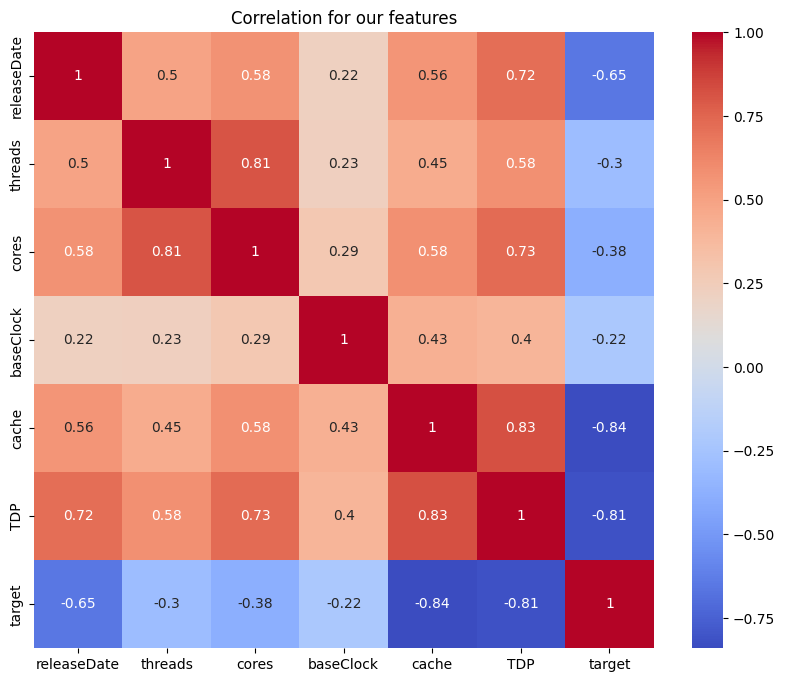
* + Confusion matrix: This matrix shows the distribution of correct and incorrect predictions for each class (AMD and Intel).
  + ROC-AUC curve: This curve helps us understand the trade-off between true positive rate and false positive rate at different classification thresholds.
  + Classification report: This report summarizes the model's performance on each class, including precision, recall, F1-score, and support (number of data points in each class).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **PRECISION** | **RECALL** | **F1-SCORE** | **SUPPORT** |
| 0 | 1.00 | 0.89 | 0.94 | 174 |
| 1 | 0.93 | 1.00 | 0.96 | 262 |
| Accuracy |  |  | 0.95 | 436 |
| macro avg | 0.96 | 0.94 | 0.95 | 436 |
| weighted avg | 0.96 | 0.95 | 0.95 | 436 |

**Table 1.1:** Classification report for evaluating the perform of a classification model

**Visualization**

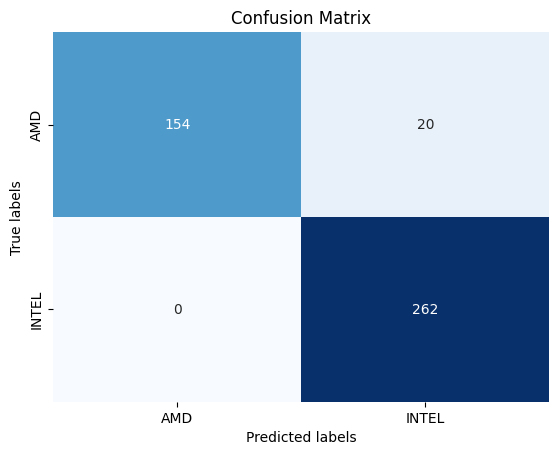
**Fig 1.1:** Data count and Target



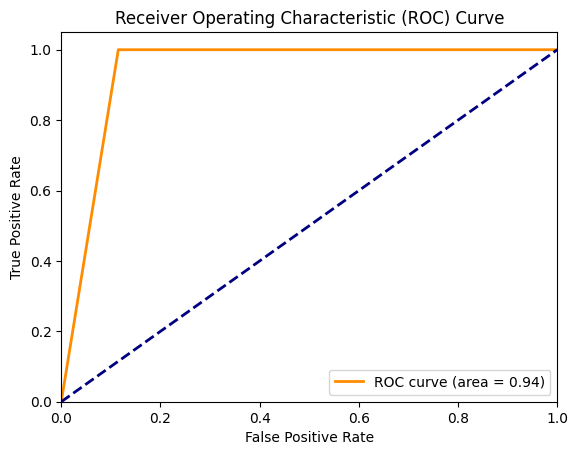
**Fig 1.2:** Feature Correlation



**Fig 1.3:** Scatter Plot to visualize the relationship between data



**Fig 1.4:** Confusion Matrix



**Fig 1.5:** ROC Curve