T.C.
FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ

Engineering Faculty
Computer Science

Data Mining Project

Name & Surname	Student ID
Muhittin AKIN	1821221006
Dilara ÇELİK	2021221042
Metin KAĞIT	1821221033

Introduction

In this project, we will predict visit reason which is order or guest. We analyze relationships between features such as entry time and date, the block and apartment visited, and the individual visitor. To compare classification methods, we train six different models—Logistic Regression, K-Nearest Neighbors, SVM, Decision Trees, Random Forest, and XGBoost, Naive Bayes and MLP (Sklearn). For each model, we perform hyperparameter tuning with five-fold cross-validation (5-CV) to find the best parameter settings. Finally, we evaluate their performance on a held-out test set using accuracy, precision, recall, F1-score, and AUC.

Data Collection and Preprocessing

We collect the data from security checkpoint of residential complex. And anonamize it to privacy. The dataset has 40367 rows and 7 columns. The columns name are:

- NO: Id. TARİH: Visit date.
- ZİYARET EDİLEN: Visited person block and apartmen no.
- TEYİT BİLGİSİ: Emty Column
- ZİYARET SEBEBİ: Visit reason
- GELİŞ SAATİ: Visit Time
- ÇIKIŞ SAATİ: Empty Colum

1. **Cleaning empty cells**
   After loading the raw data from Excel with pandas, all empty cells (NaN) were removed.

2. **Renaming columns**
   The original column headers were renamed to be more meaningful and consistent (e.g. ZİYARET SEBEBİ → visit_reason).

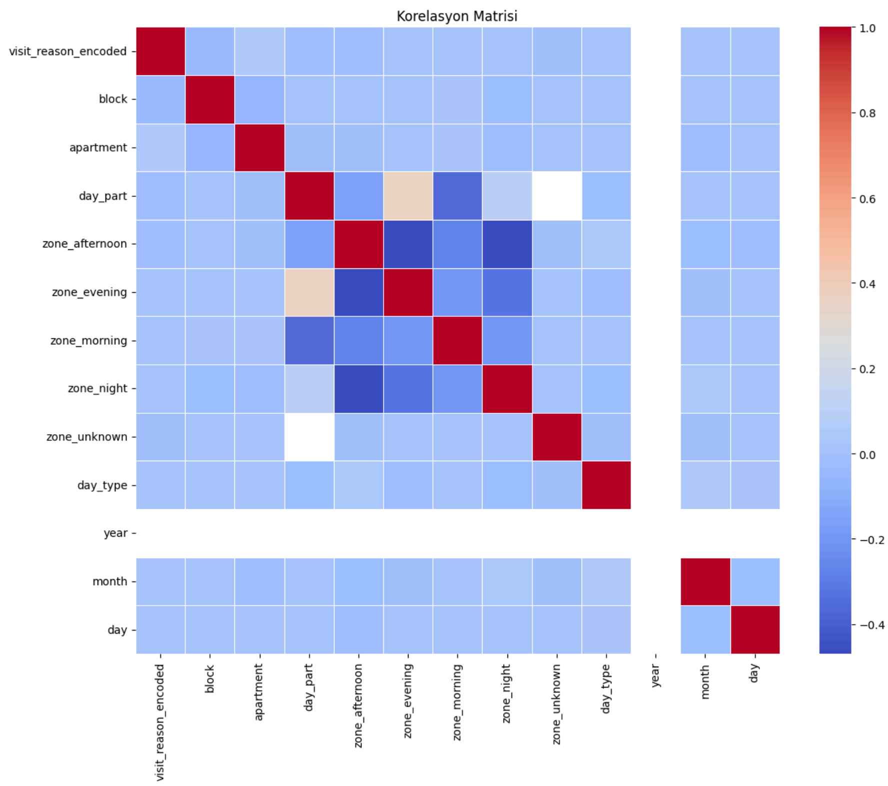
3. **Removing rows with missing values**
   Any row containing a missing value in any column was dropped,
   because these gaps could not be logically or numerically imputed and might harm model performance.

4. **Feature Engineering**

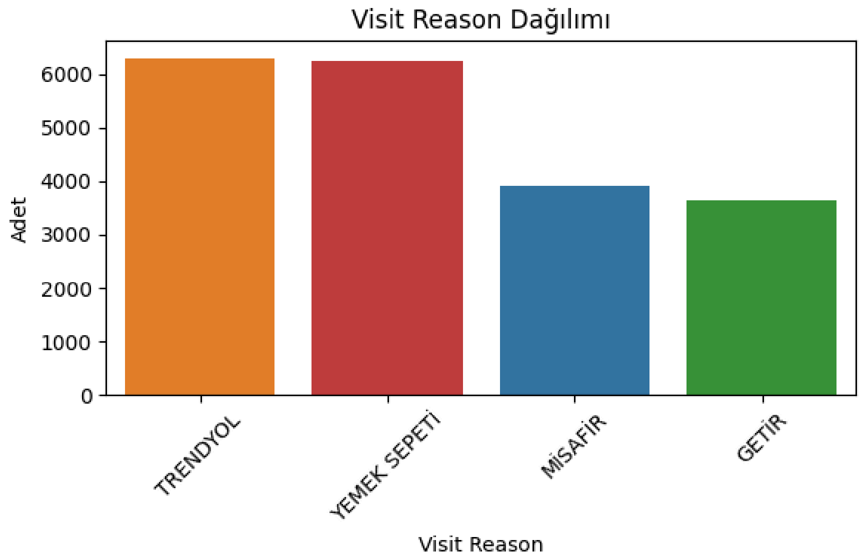
Since the original dataset had relatively few features, we created new variables:

- **visit_reason_encoded:** Encoded version of visit_reason that include most repeated 4 unique value.
- **visit_reason_Binary:** Encoded version of visit_reason that include 2 value is_order 1 or not 0.
- **block & apartment:** Separate the visited column into two column, one of them just encoded block name and the other one is apartment number.
- **day_part:** The arrival_time column was split into five time-of-day categories—unknown, morning, afternoon, evening, and night. For each record, the column corresponding to its arrival slot is set to 1, and the others are set to 0.
- **day_type:** A new column day_type was created to classify each record as either weekday or weekend.
- **day&month&year:** date column separate 3 part which are day, month and year.

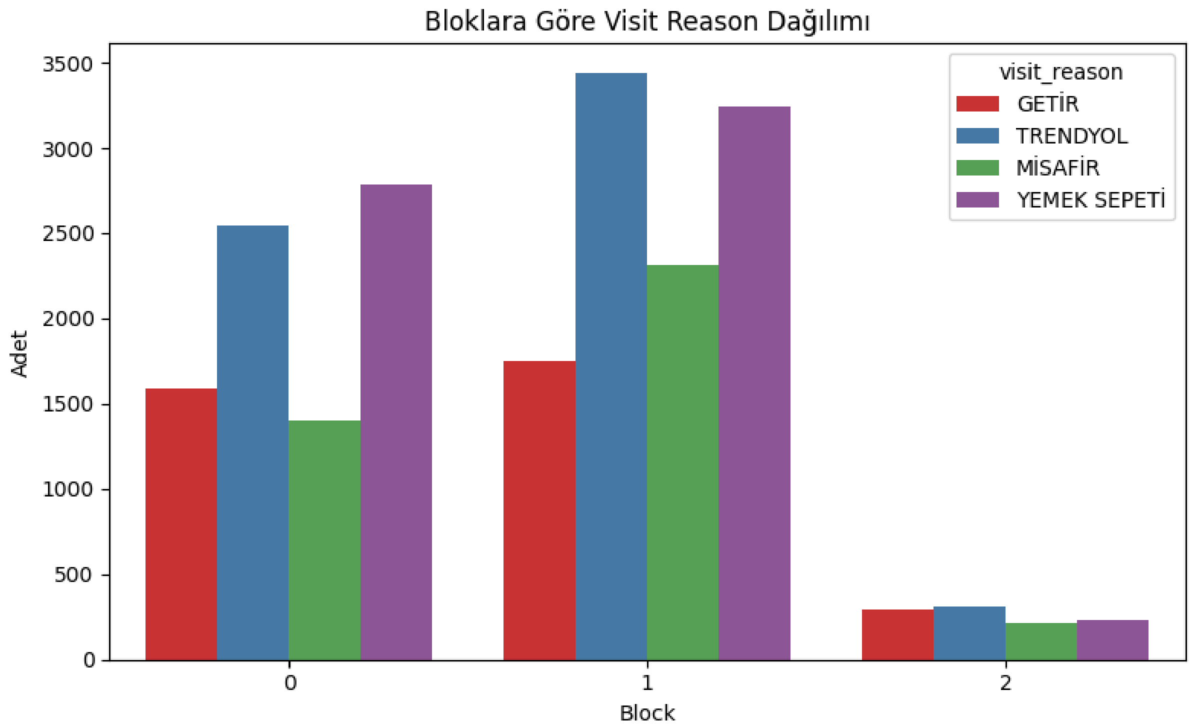
Data Visualization



Since there is no linear correlation, the chosen models should be those that handle non-linear relationships well, such as Random Forest.

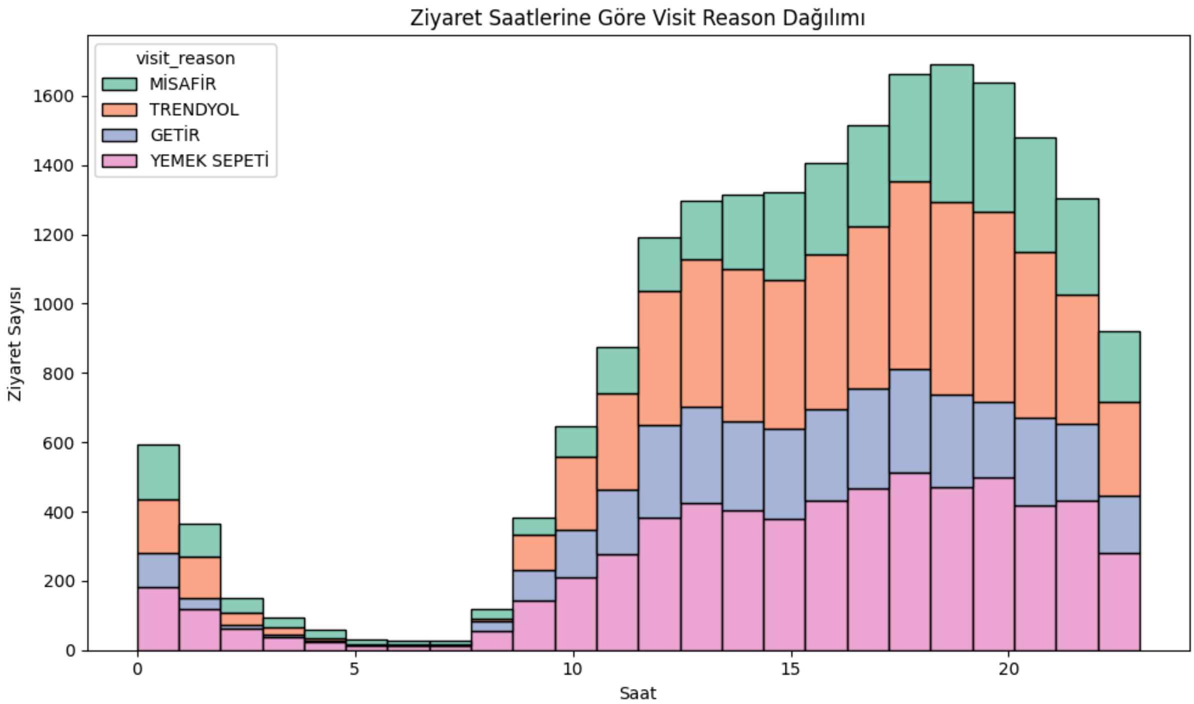


The graph show the amount of visit from most bigger 4 value.



0:A, 1:B, 2:C

The highest number of flats are B, A and C, respectively. It is observed that this situation also affects the number of visitors.



As we seen, there is a higher number of visitors in the afternoon and evening hours.

Model Construction

In the model construction phase, we implemented six classification algorithms—Logistic Regression, K-Nearest Neighbors, Support Vector Machine, Decision Tree, Random Forest, XGBoost, Naive Bayes and MLP (Sklearn) — within scikit-learn pipelines and optimized their hyperparameters via GridSearchCV with cross-validation to identify the best performer for predicting visit reasons.

Firstly we trained our model without cross validation and tuning and the result can see in the table below.

Model	Set	Accuracy	Error	Precision	Recall	F1-score	AUC
Logistic Regression	Train	0.5296570898980537	0.4703429101019463	0.5310679611650485	0.5069508804448564	0.5187292555713608	0.5410894785148054
Logistic Regression	Test	0.5207556549838429	0.4792443450161571	0.8265204386839482	0.5120444718962323	0.6323417238749046	0.544248435182526
KNN	Train	0.8604031510658017	0.13959684893419833	0.8716845878136201	0.845227062094532	0.8582519703564286	0.9426201604259294
KNN	Test	0.6716380810340542	0.3283619189659458	0.8189683860232945	0.7600370599135269	0.7884030113727375	0.5628387815078113
SVM	Train	0.7197636700648748	0.28023632993512515	0.6667643439020102	0.8786685202347853	0.758188664156476	0.8036465567123416

SVM	Test	0.7335321899080288	0.2664678100919712	0.8135495078170237	0.8678196417541693	0.8398087268380156	0.5731766089785706
Decision Tree	Train	0.9960225517454433	0.003977448254556704	0.9995333281480905	0.9925084955205438	0.9960085254795582	0.9999681957792436
Decision Tree	Test	0.7355207556549839	0.26447924434501613	0.8413944723618091	0.8273625694873379	0.834319526627219	0.5929497645397214
Random Forest	Train	0.9960225517454433	0.003977448254556704	0.9944568481022403	0.9976058078467717	0.9960288391101515	0.9999411454902092
Random Forest	Test	0.7834949043002735	0.21650509569972654	0.827204865910976	0.9240271772699197	0.8729394602479942	0.6648159003552558
XGBoost	Train	0.8687828236021007	0.13121717639789932	0.9557167398358465	0.773401297497683	0.8549474942371724	0.974339558991652
XGBoost	Test	0.7101665423813075	0.2898334576186925	0.8957219251336899	0.7242124768375541	0.8008879781420765	0.7480718616115162
Gaussian NB	Train	0.5968489341983317	0.40315106580166826	0.5909882455376578	0.6290546802594995	0.6094276094276094	0.628945357724236
Gaussian NB	Test	0.5973154362416108	0.40268456375838924	0.8275303643724696	0.6312538604076591	0.716187806587246	0.5571446556221304
MLP (Sklearn)	Train	0.7494593759654	0.2505406240346	0.7051575203252033	0.857429718875502	0.7738742506622055	0.8307336842438788
MLP (Sklearn)	Test	0.7216007954262987	0.27839920457370126	0.8155542312276519	0.8452748610253242	0.8301486199575372	0.5731701175924433

Logistic Regression and Gaussian NB both show low performance, with training and test accuracies around 52–60% and AUC scores of about 0.55. KNN overfits, reaching 86% accuracy on training but dropping to 67% on test. SVM (72–73% accuracy, F1 ~84%, AUC ~0.57) and MLP (75–72% accuracy, F1 ~83%, AUC ~0.57) stay balanced on both training and test, with almost no overfitting.

Decision Tree (99% training, 73% test) and Random Forest (99% training, 78% test) also overfit, but Random Forest gives the best test results with 78% accuracy and F1 ~87%. XGBoost (87% training, 71% test) controls overfitting better and, with a test AUC of ~0.75, offers the most balanced class separation.

After that, we train the models with hyper parameter optimization and cross validation to improve model performance and stability.

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**	**AUC**
Logistic Regression	**0.523743**	**0.824246**	**0.518966**	**0.636596**	**0.537356**
KNN	**0.687833**	**0.824358**	**0.777895**	**0.800435**	**0.57098**
SVM	**0.75317**	**0.816198**	**0.894848**	**0.853701**	**0.5752**
Decision Tree	**0.737358**	**0.847033**	**0.822131**	**0.834384**	**0.605229**
Random Forest	**0.79459**	**0.832988**	**0.931546**	**0.879513**	**0.684349**
XGBoost	**0.706678**	**0.89694**	**0.718091**	**0.797573**	**0.755434**
Gaussian NB	**0.592462**	**0.827852**	**0.623256**	**0.711097**	**0.561607**
MLP (Sklearn)	**0.731441**	**0.819947**	**0.853825**	**0.836481**	**0.582883**

According to these results, the best model is Random Forest, but XGBoost has the highest AUC score of all. Therefore, if you prioritize overall accuracy and recall, choose Random Forest; if you care more about ranking and discrimination ability, choose XGBoost.

Conclusion

Our comprehensive evaluation compared various classification models using both a single train/test split and 5-fold cross-validation metrics, and we also tuning parameters by hyper parameter optimization and analyzed each model's AUC from the last fold's ROC curves. As a result, The less success model is Logistic Regression because Logistic Regression good to learned from linear relations our data is not contain linear relations. Also, XGBoost emerges as the most balanced model overall, while Random Forest ranks second for having the highest recall and test accuracy. If your goal is to maximize overall discrimination power and AUC, choose XGBoost; if you need to avoid missing any positive cases, Random Forest is the better choice.