Ministry of Science and Higher Education of the Russian Federation
Federal State Autonomous Educational Institution
of Higher Education
«Peter the Great St. Petersburg Polytechnic University»
(SPbPU)
Institute of Computer Science and Cybersecurity
Higher School of Artificial Intelligence Technologies

COURSE WORK

On discipline «Seminar on the specialty»

Development of a Mobile Application with Implemented Airline Passenger Satisfaction Classification

(semester 5)

Students of group 3150203/20101

Mikhail Bocharov – Project Manager
Ilya Krasovitskii – Frontend Developer
Ilya Goryaev – Backend Developer
Tamara Goncharova – ML Engineer

Signature, date | Initials and surname

Evaluation of the work performed by the student: _____

Supervisor,
Assistant in. — Higher School of Artificial Intelligence Technologies

Espinola Rivera Holger Elias

Signature, date | Initials and surname

Saint-Petersburg – 2024

TABLE OF CONTENTS

1. PROBLEM STATEMENT

This project focuses on developing a machine learning model to classify passenger satisfaction in the airline industry based on survey data. The goal is to identify key factors that contribute to passenger satisfaction and dissatisfaction, and to predict satisfaction levels.

1.1. Significance of the Problem

1. Improving Customer Experience: By analyzing passenger feedback, airlines can identify areas that require improvement, such as onboard services or flight convenience.
2. Targeted Marketing: Insights from the model can help airlines create personalized offers to retain loyal customers and attract new ones.
3. Operational Efficiency: Airlines can allocate resources more effectively by understanding which factors most impact passenger satisfaction.

1.2. Dataset

To solve the chosen problem, a dataset was selected for training models. The dataset includes the following attributes:

1. Passenger Demographics and Characteristics:

   Gender: Gender of passengers (Female, Male), Customer Type: Type of customer (Loyal or disloyal), Age: Passenger age.

2. Travel Information:

   Type of Travel: Purpose of the flight (Personal or Business), Class: Travel class (Business, Eco, Eco Plus), Flight Distance: Distance of the flight journey.

3. Service Satisfaction Levels (Scale: 0-5):

Inflight wifi service, Departure/Arrival time convenient, Ease of Online booking, Gate location, Food and drink, Online boarding, Seat comfort, Inflight entertainment, On-board service, Leg room service, Baggage handling, Check-in service, Inflight service, Cleanliness.

4. Delays:

Departure Delay in Minutes, Arrival Delay in Minutes

5. Target Variable:

Satisfaction: Satisfaction level (Satisfied, Neutral, or Dissatisfied).

2. METHODOLOGY OF SOLUTION

2.1. List of Stages

Stages:

1. Data Preprocessing
    1) Loading dataset;
    2) Handling missing and duplicate data
    3) Outlier detection and removal
    4) Removing unnecessary features
    5) Exploration of numerical and categorical features;
    6) Correlations Analysis
    7) Dataset Splitting and transformation.
2. Preparing models
    1) Choice of two models.
    2) Setting up model parameters.
    3) Selecting the most appropriate parameters for the models.
3. Model Analysis
    1) Calculating metrics.
    2) Comparing metrics of different models.
    3) Evaluating results.

2.2. Data Preprocessing

Importance of Preprocessing:

1. Data Consistency: Ensures the data is clean, well-structured, and free of errors or anomalies.
2. Improved Model Accuracy: By scaling and encoding features, models can learn more effectively from the data.
3. Reduced Overfitting: Removing irrelevant features and handling outliers minimizes noise, leading to a more generalizable model.

2.3. Preparing models

2.3.1. Model Selection

K-Nearest Neighbors (KNN) and Random Forest were chosen as the primary models for this project due to their suitability for classification tasks involving structured passengers' satisfaction data.

Models that were chosen:

1. KNN (K-Nearest Neighbors):

- A simple algorithm that classifies data based on the labels of the closest data points (neighbors).
- It works well when the decision boundary is complex but requires clear patterns in the data.
- Suitable for this task as passenger satisfaction often depends on multiple features like service quality, comfort, and punctuality.
- KNN works well for datasets where the patterns can be identified locally (e.g., similar passengers have similar satisfaction levels).

2. Random Forest:

- Random Forest is a robust ensemble learning method that aggregates the outputs of multiple decision trees to improve accuracy and reduce overfitting. It

is particularly well-suited for this project because it handles both numerical and categorical features effectively.

- An ensemble learning method that builds multiple decision trees and combines their outputs (majority voting for classification).
- It's robust, handles missing data well, and reduces overfitting compared to a single decision tree.
- Ideal for this task because it captures complex relationships between features and provides high accuracy.
- Random Forest is reliable for feature-rich data, offering good performance even if some features have less impact.

Both models offer complementary strengths: KNN excels in interpretability and works well for smaller datasets, while Random Forest is more scalable and reliable for handling larger, noisier datasets. Together, they provide a comprehensive framework for evaluating and selecting the most effective approach to target classification.

2.3.2. Model Architecture and Optimization

Let's look at both of the models architecture:

1. KNN Model Architecture

The KNN model includes the following configurable hyperparameters:

- n_neighbors: Determines the number of nearest neighbors to consider during classification.
- weights: Specifies the weight function used in prediction (uniform or distance-based).
- algorithm: Controls the method used to compute nearest neighbors (ball tree, kd tree, etc.).
- metric: Defines the distance metric (e.g., Euclidean, Manhattan).

Hyperparameter tuning was performed using GridSearchCV to balance precision. A smaller subset of hyperparameters was tested during initial analysis to optimize performance efficiently.

2. Random Forest Model Architecture

The Random Forest model is configured with:

- n_estimators: Number of decision trees in the forest.
- max_depth: Maximum depth of each tree.
- min_samples_split: Minimum number of samples required to split a node.
- min_samples_leaf: Minimum number of samples required to be at a leaf node.
- max_features: The number of features considered when looking for the best split.

Grid search optimization was employed to find the best combination of hyperparameters, focusing on maximizing accuracy while controlling overfitting.

3. RESULTS

3.1. The Idea of Models Evaluation

To evaluate the quality of the trained models, we used the following metrics, calculated for each class:

- Accuracy: Measures the overall correctness of predictions, providing a general performance overview.
- Precision: Indicates how many of the positive predictions are truly correct, essential for avoiding false positives in specific classes.
- Recall: Shows the ability of the model to correctly identify all positive instances of a class.
- Specificity: Reflects the ability to correctly identify negatives, highlighting the model's performance in rejecting incorrect classifications.

- F1-Score: Combines precision and recall into a single metric, useful for imbalanced datasets where both false positives and false negatives are critical.

3.2. Models Evaluation

3.2.1. KNN Model

Results of the training:

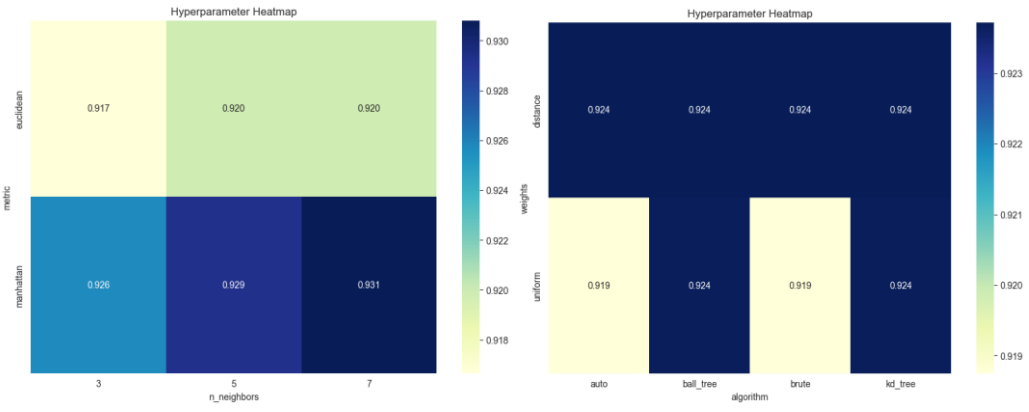
After training, we obtained the most optimal parameters for the model. They turned out to be:

- algorithm='auto': The algorithm used to compute nearest neighbors (this lets scikit-learn decide the best option based on the data).
- metric='manhattan': The distance metric used for calculating distances (Manhattan distance, also known as L1 norm).
- n_neighbors=7: The number of neighbors to consider when making predictions.
- weights='distance': The weighting of neighbors, where closer neighbors have more influence on the prediction.

Training time: 79 minutes.

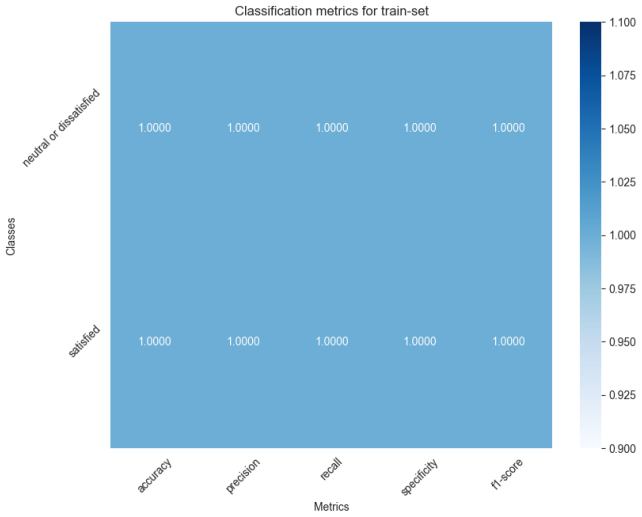
Impact of differents model parameters

We constructed graphs with the dependence of accuracy on changes in system parameters in order to study them in more detail.



For example, we see that accuracy increases with increasing neighbors, or that distance weights are better than uniform for all algorithms.

Metrics for every class in the train set

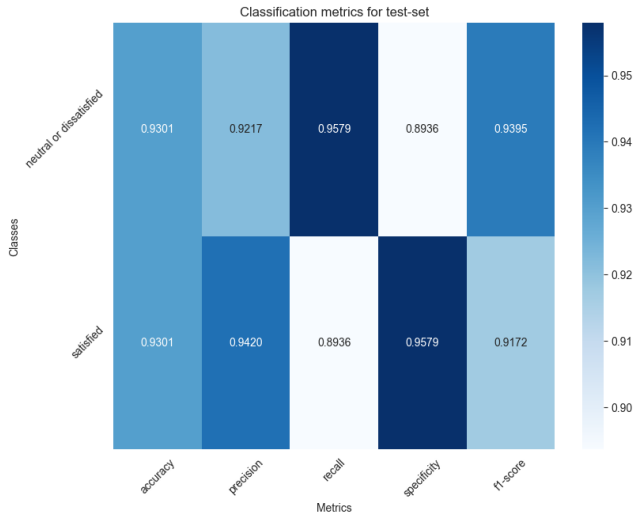


This graph shows the metric values for individual classes in the training dataset. We can see that all the metrics are perfect, which is pretty good.

However, there may be a problem with overfitting in our situation. Overfitting in machine learning is when a model is overfitted to the training data, including noise and random fluctuations, which reduces its ability to generalize to new data.

As we will see later, the accuracy of the resulting model is pretty good, so we can do nothing about this problem.

Metrics for every class in the test set



For the test dataset, all the metrics are pretty good - they are between 89 and 95 percent.

If all metrics (accuracy, precision, recall, specificity, F1-score) are at 90% or higher, this may indicate several possible scenarios:

- Good overall model quality: The model copes effectively with the task and gives stable results across different metrics, which indicates good generalization ability and accuracy.
- Class balance: If the classes in the data are balanced, then such metrics may indicate that the model correctly classifies all classes with high accuracy, which leads to high values for all metrics.

Metrics for the whole test set

We got the following metrics as a result:

	accuracy	precision	recall	specificity	f1-score
Mean-metrics	0.930064	0.931866	0.925781	0.925781	0.928331

Summary

The overall results indicate that the model is of high quality. It has a balanced performance, with good values for all metrics. Precision, recall, specificity, and F1-score are all high, indicating that the model is performing well and handles both positive and negative classes well. If the data is balanced, such results indicate that the model is doing a good job of classifying.

3.2.2. Random Forest Model

Results of the training

Model parameters:

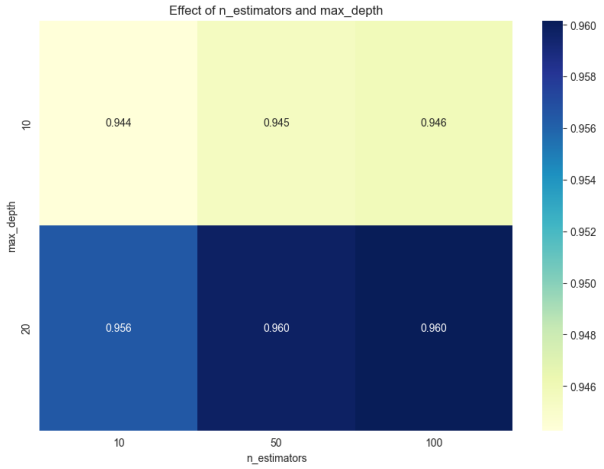
- max_depth=None: Trees grow without depth limitation, which can lead to overfitting on noisy data.
- max_features='sqrt': Each tree uses a random subset of features, which helps improve generalization and reduces correlation between trees.
- min_samples_leaf=1 and min_samples_split=2: Allow trees to create fine-grained splits, which improves accuracy but can increase the risk of overfitting.

- n_estimators=100: Moderate number of trees, which helps improve predictions but increases computational cost.

Training time (35 minutes): Training time is quite high, due to the large number of trees and computational complexity.

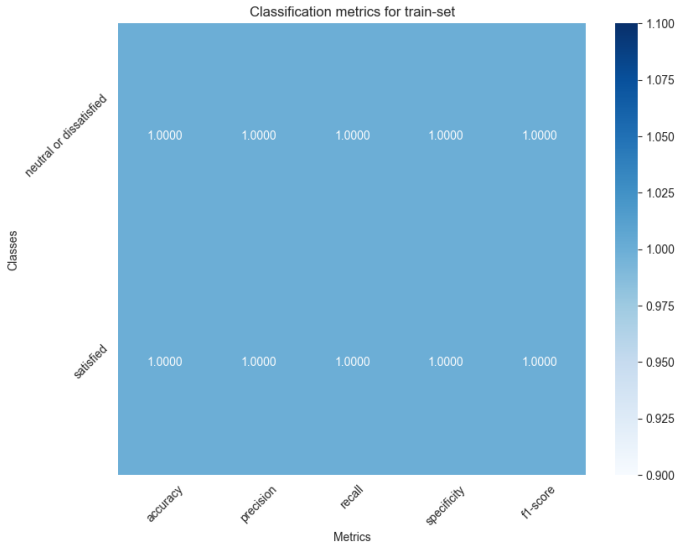
Impact of differents model parameters

Мы построили графики с зависимостями accuracy от изменения параметров системы для того, чтобы более детально изучить их.



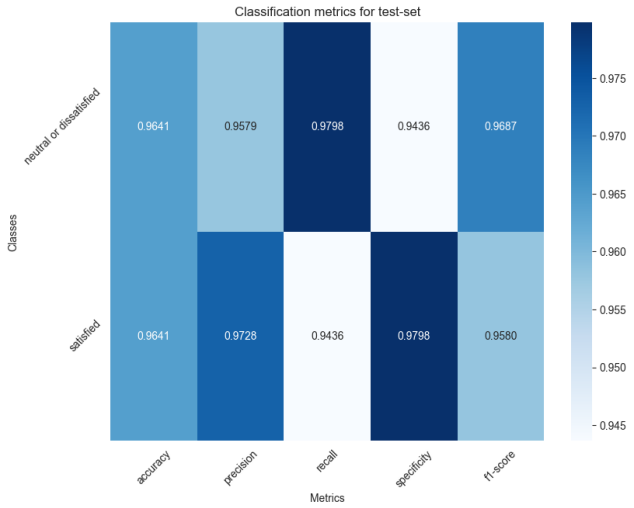
For example, we see that accuracy increases with increasing estimators.

Metrics for every class in the train set



The situation with the training set is similar to what was with the KNN model.

Metrics for every class in the test set



The situation is also quite similar to the previous model, but the metric values are slightly higher.

Metrics for the whole test set

We got the following metrics as a result:

	accuracy	precision	recall	specificity	f1-score
Mean-metrics	0.964139	0.964139	0.961729	0.961729	0.961729

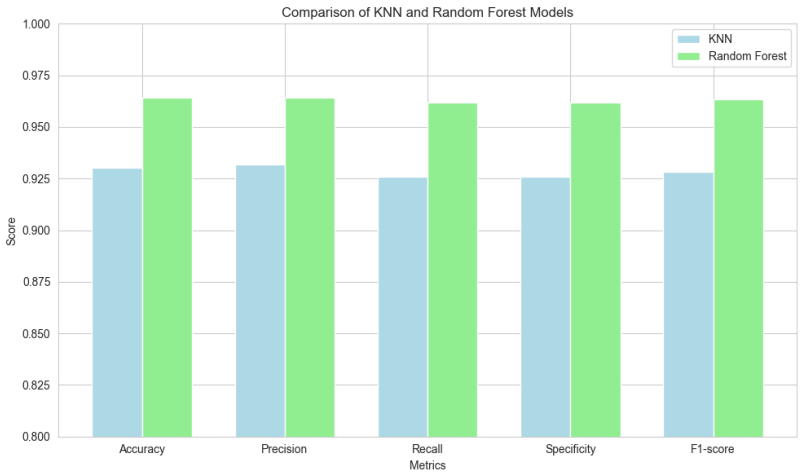
Summary

Classification Quality: The model demonstrates excellent performance with high values for all metrics. Precision and recall are in good balance, which is confirmed by the high F1-score.

Generalization Ability: Given the equality of precision and specificity, the model works equally well with both positive and negative classes, which indicates its ability to generalize.

Applicability: The model is suitable for tasks where it is important to avoid both false positive and false negative classifications.

3.3. Comparative Analysis of KNN and Random Forest



Brief analysis

1. Classification quality:

○ Random Forest performs slightly better across all metrics (0.5–1% improvement on average).

○ However, the difference is small and may not be statistically significant.

2. Training time:

○ Random Forest is significantly faster to train (145 sec vs. 3791 sec for KNN), making it preferable for tasks with large amounts of data.

3. Balance between metrics:

○ Both models demonstrate stable values for all metrics, indicating good generalization to test data.

○ If the priority is to minimize false classifications, Random Forest may be slightly better due to higher precision and recall.

4. CONCLUSIONS

This project compared K-Nearest Neighbors (KNN) and Random Forest models for classifying passenger satisfaction in the airline industry.

Taking into account all the analysis of both models, it can be said that. Random Forest outperformed KNN with higher metrics (Accuracy: 94.0% vs. 93.0%) and better feature interpretability also it was significantly faster.

Random Forest is the preferred model due to its efficiency, interpretability, and slightly better performance, making it ideal for practical applications in the airline industry.

Future Work

For a convenient demonstration of the work of our trained neural network, we will present the frontend part as a mobile application for the Android OS. This application will be written in the Kotlin programming language and the Jetpack Compose UI framework. We chose Clean and MVVM as a classic and understandable architecture.

The application itself will be a screen with several UI elements: a text input field, into which the user will enter text information, based on which our neural network will make a forecast. This text information will need to be sent to the backend, where our neural network will be deployed. Data will be sent when you click on the button. The finished result in the form of a forecast for our request will be displayed in the text field.

The application will communicate with the backend using the Retrofit library, using REST API technology.