

# **Ramsay Data Insights and Model Development**

# Contents

|          |   |          |
|----------|---|----------|
| <b>1</b> | <b>Data Exploration and Preparation</b>                     | <b>3</b> |
| 1.1      | Data Problems . . . . .                                     | 3        |
| 1.2      | Feature engineering . . . . .                               | 4        |
| <b>2</b> | <b>Part 2: Data Analysis and Visualisation</b>              | <b>5</b> |
| 2.1      | Total charge visualization . . . . .                        | 5        |
| <b>3</b> | <b>Part 3: SQL query</b>                                    | <b>5</b> |
| <b>4</b> | <b>Part 4: Strategic Insights and Recommendations</b>       | <b>8</b> |
| 4.1      | Charge Discrepancies Among AR-DRG Codes . . . . .           | 8        |
| 4.2      | Variability in TotalCharges by PrincipalDiagnosis . . . . . | 8        |
| <b>5</b> | <b>Part 5: Model Development</b>                            | <b>8</b> |
| <b>6</b> | <b>Part 6: MLOps and Deployment</b>                         | <b>9</b> |

# 1 Data Exploration and Preparation

## 1.1 Data Problems

### 1.1.1 Missing Data

**Description:** Several columns have a significant amount of missing values, with the percentages of missing values being quite high. For instance, columns like UnplannedTheatreVisit, InfantWeight, Readmission28Days, PalliativeCareStatus, and HoursMechVentilation all have missing data rates exceeding 90%, with some as high as 96%. This level of missing data can drastically affect the quality of insights drawn from the analysis and potentially introduce bias or reduce statistical power if not handled properly.

**Impact:** Missing data may skew results and lead to inaccurate or biased analysis, especially if certain patterns in the missingness are related to important factors in the analysis. Decisions need to be made about how to handle this (e.g., imputation, dropping columns, or rows).

**Solution:** For this project, I will use an imputation approach that fills missing values with the median value. The median value is filled based on each DRG.

|    | Feature name          | Missing Values | Percentage (%) |
|----|-----------------------|----------------|----------------|
| 0  | CCU_Charges           | 17167          | 57.22          |
| 1  | ICU_Charge            | 17138          | 57.13          |
| 2  | TheatreCharge         | 17164          | 57.21          |
| 3  | PharmacyCharge        | 16735          | 55.78          |
| 4  | ProsthesisCharge      | 17151          | 57.17          |
| 5  | OtherCharges          | 17181          | 57.27          |
| 6  | BundledCharges        | 17106          | 57.02          |
| 7  | UnplannedTheatreVisit | 28990          | 96.63          |
| 8  | InfantWeight          | 28475          | 94.92          |
| 9  | Readmission28Days     | 28989          | 96.63          |
| 10 | HoursMechVentilation  | 28531          | 95.10          |
| 11 | PalliativeCareStatus  | 28978          | 96.59          |

### 1.1.2 Outlier

The figures 1 and 2 present boxplots for charge-related features before and after the removal of outliers, respectively. In the first figure, the boxplot highlights key summary statistics, such as the median, interquartile range (IQR), and potential outliers for features like TheatreCharge, ProsthesisCharge, and BundledCharges. These boxplots clearly reveal the presence of extreme values that deviate significantly from the bulk of the data, which is especially noticeable for the aforementioned variables.

Outliers of this magnitude can have a considerable impact on data analysis and model training. They may originate from various sources, including data entry errors, uncommon patient treatments involving complex and high-cost procedures, or rare occurrences that require special medical attention. Regardless of their origin, these outliers can skew summary statistics and negatively affect the performance of machine learning models, causing models to fit to noise rather than underlying patterns in the data.

To address these extreme values, the Interquartile Range (IQR) method was used to identify and filter out outliers. This method flags data points as outliers if they fall beyond 1.5 times the IQR from the first and third quartiles. The second boxplot (Figure 2 illustrates the same charge features after the removal of these outliers. With the extreme values removed, the spread of the data becomes more compact and representative of the majority of the dataset.

Removing outliers helps to reduce their undue influence on statistical metrics, such as the mean, and it improves the robustness of machine learning models by allowing them to focus on typical patterns rather than rare anomalies. This not only enhances model accuracy but also ensures that the models are more generalizable to future data. Additionally, by eliminating these extreme values, we improve the interpretability of the results, as the models are no longer biased by a small number of highly unusual cases. This preprocessing step is crucial for developing reliable and realistic predictive models for hospital charges.

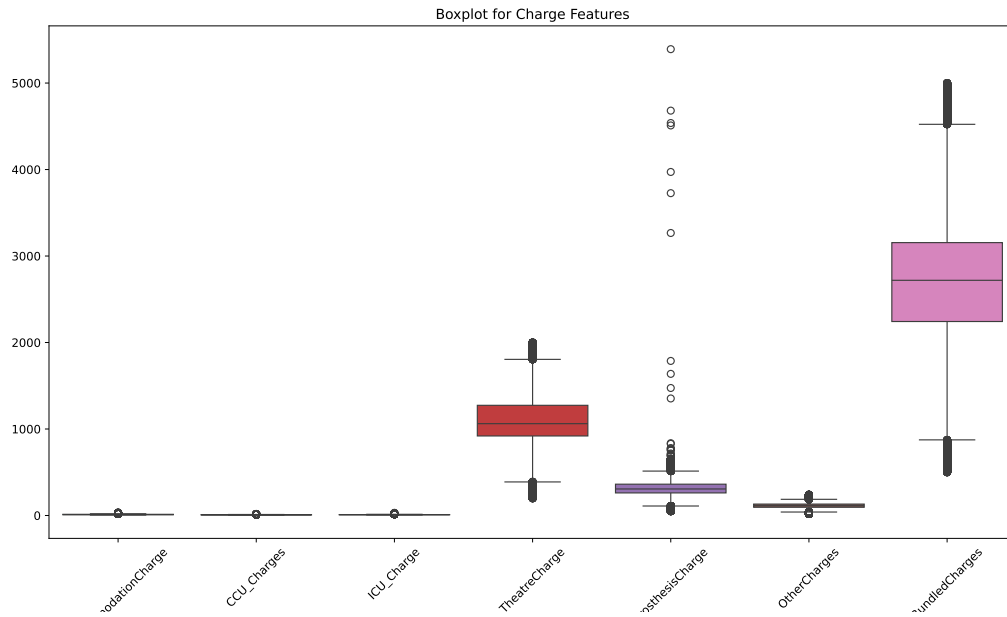


Figure 1: Original Charge Features

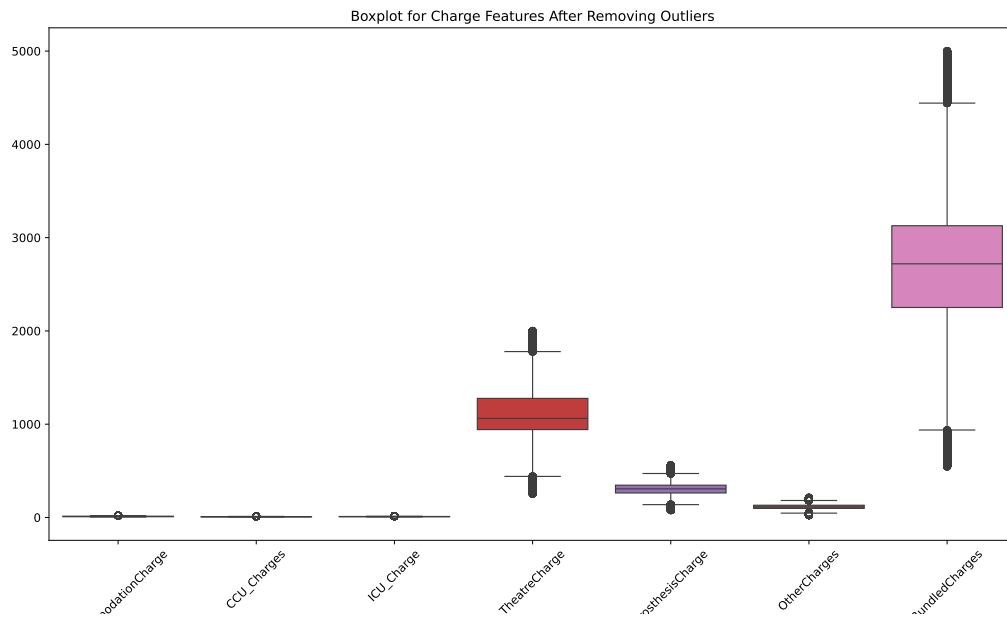


Figure 2: Charge Features After Removing Outliers

## 1.2 Feature engineering

In the provided dataset, several valuable features have been created that can significantly contribute to analysis or predictive modeling:

- **Length of Stay:** This feature calculates the number of days between the patient's admission and separation date. It is essential for understanding the duration of hospital stays, which can be indicative

of the severity of a patient's condition, resource usage, or recovery patterns. This feature is crucial for predicting hospital costs, bed management, and resource allocation.

- **Weekend Stay:** This binary feature indicates whether a patient was admitted during a weekend (Friday or later). The weekend admission can be correlated with different patterns in resource use, staffing, and outcomes, as healthcare services may differ during weekends. This can be useful for understanding whether weekend admissions affect patient outcomes or hospital operations.
- **ICU/CCU Use:** This binary feature indicates whether a patient utilized ICU (Intensive Care Unit) or CCU (Coronary Care Unit) services. This feature is critical in identifying patients who required critical care and can be used for predicting the severity of cases, resource-intensive admissions, or patient outcomes. Additionally, it can help in financial forecasting and resource allocation related to critical care facilities.
- **Age Group:** The age group feature categorizes patients into bins such as "Child," "Young Adult," "Adult," and "Elderly," providing an easy way to segment patients based on age. Age is often a strong predictor in medical outcomes, as younger and older populations might have distinct healthcare needs. This feature is useful in demographic analysis, risk prediction, and patient profiling for different treatment approaches.

These features offer insightful opportunities for predictive modeling, such as forecasting hospital resources, predicting patient outcomes, or performing cost analysis based on patient characteristics. By incorporating these features into models, we can improve predictions and tailor healthcare services to specific patient groups.

## 2 Part 2: Data Analysis and Visualisation

### 2.1 Total charge visualization

In this analysis, we examine the top 10 Diagnosis-Related Groups (DRGs) that accrue the largest total charges in the hospital dataset. As shown in both the table and the bar plot, DRG002, DRG003, and DRG001 are the three DRGs with the highest total charges, each surpassing 20 million dollars. However, it's important to note that DRGs such as Q62A and L62C have comparatively higher average charges per case, despite having lower total charges. This suggests that while some DRGs might have fewer total admissions, the cost per admission is significantly higher, potentially indicating more resource-intensive treatments or complex cases. Understanding the DRGs that contribute the most to hospital expenses help identify areas for cost optimization and resource allocation.

| AR-DRG | total_charges | average_charges |
|--------|---------------|-----------------|
| DRG002 | 19287283.31   | 4258.62         |
| DRG003 | 18934732.45   | 4203.98         |
| DRG001 | 18736118.89   | 4240.86         |
| Q62A   | 164765.64     | 5492.19         |
| C63B   | 151123.98     | 5211.17         |
| B76A   | 141584.69     | 4882.23         |
| I08B   | 139884.28     | 4662.81         |
| L62C   | 139054.44     | 4485.63         |
| T61B   | 133494.26     | 4944.23         |
| F74A   | 132585.32     | 5524.39         |

## 3 Part 3: SQL query

Write an SQL query to calculate the total and average admissions for each month over the last two years. Include the month and year in the results.

```
1 SELECT
2     strftime('%Y', AdmissionDate) AS year,
3     strftime('%m', AdmissionDate) AS month,
4     COUNT(*) AS total_admissions,
```

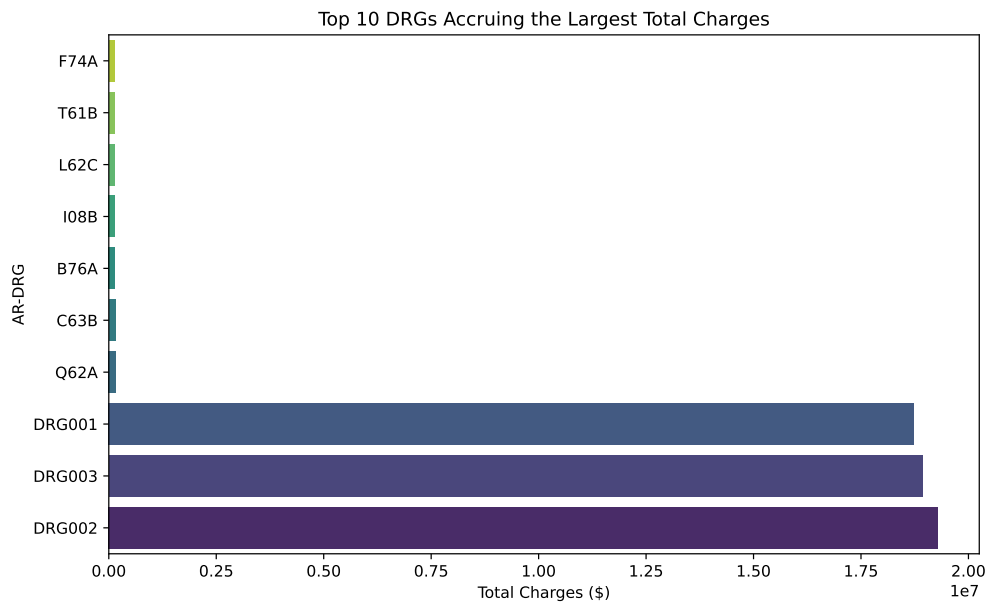


Figure 3: Total Charge by DRGs

```

5      AVG(COUNT(*)) OVER (PARTITION BY strftime('%Y', AdmissionDate), strftime('%m', AdmissionDate)) AS average_admissions
6 FROM admissions_table
7 WHERE AdmissionDate >= date('now', '-2years')
8 GROUP BY year, month
9 ORDER BY year DESC, month DESC;

```

Write an SQL query to analyse the distribution of TotalCharges by PrincipalDiagnosis and Sex. Use percentiles to describe the distribution.

```

1 WITH RankedCharges AS (
2     SELECT
3         PrincipalDiagnosis,
4         Sex,
5         TotalCharges,
6         ROW_NUMBER() OVER (PARTITION BY PrincipalDiagnosis, Sex ORDER BY
7             TotalCharges) AS row_num,
8         COUNT(*) OVER (PARTITION BY PrincipalDiagnosis, Sex) AS total_rows
9     FROM
10         admissions_table
11     WHERE
12         TotalCharges IS NOT NULL
13 )
14 SELECT
15     PrincipalDiagnosis,
16     Sex,
17     MIN(TotalCharges) AS min_total_charges,
18     MAX(TotalCharges) AS max_total_charges,
19     AVG(CASE WHEN row_num = (total_rows + 1) / 4 THEN TotalCharges END) AS
20     p25_total_charges,
21     AVG(CASE WHEN row_num = (total_rows + 1) / 2 THEN TotalCharges END) AS
22     median_total_charges,
23     AVG(CASE WHEN row_num = 3 * (total_rows + 1) / 4 THEN TotalCharges END) AS
24     p75_total_charges,
25     COUNT(*) AS total_records

```

|    | year | month | total_admissions | average_admissions |
|----|------|-------|------------------|--------------------|
| 0  | 2024 | 07    | 1129             | 1129.0             |
| 1  | 2024 | 06    | 1157             | 1157.0             |
| 2  | 2024 | 05    | 1213             | 1213.0             |
| 3  | 2024 | 04    | 1127             | 1127.0             |
| 4  | 2024 | 03    | 1134             | 1134.0             |
| 5  | 2024 | 02    | 1045             | 1045.0             |
| 6  | 2024 | 01    | 1116             | 1116.0             |
| 7  | 2023 | 12    | 1125             | 1125.0             |
| 8  | 2023 | 11    | 1062             | 1062.0             |
| 9  | 2023 | 10    | 1165             | 1165.0             |
| 10 | 2023 | 09    | 1058             | 1058.0             |
| 11 | 2023 | 08    | 1101             | 1101.0             |
| 12 | 2023 | 07    | 1129             | 1129.0             |
| 13 | 2023 | 06    | 1120             | 1120.0             |
| 14 | 2023 | 05    | 1153             | 1153.0             |
| 15 | 2023 | 04    | 1116             | 1116.0             |
| 16 | 2023 | 03    | 1116             | 1116.0             |
| 17 | 2023 | 02    | 1035             | 1035.0             |
| 18 | 2023 | 01    | 1104             | 1104.0             |
| 19 | 2022 | 12    | 1111             | 1111.0             |
| 20 | 2022 | 11    | 1115             | 1115.0             |
| 21 | 2022 | 10    | 381              | 381.0              |

```

22 FROM
23     RankedCharges
24 GROUP BY
25     PrincipalDiagnosis , Sex
26 ORDER BY
27     PrincipalDiagnosis , Sex;

```

|       | PrincipalDiagnosis | Sex | min_total_charges | max_total_charges | p25_total_charges | median_total_charges | p75_total_charges | total_records |
|-------|--------------------|-----|-------------------|-------------------|-------------------|----------------------|-------------------|---------------|
| 0     | A00.1              | F   | 4635.157287       | 4635.157287       | NaN               | 4635.157287          | 4635.157287       | 1             |
| 1     | A00.2              | F   | 4651.620961       | 4651.620961       | NaN               | 4651.620961          | 4651.620961       | 1             |
| 2     | A00.2              | M   | 3702.404329       | 3702.404329       | NaN               | 3702.404329          | 3702.404329       | 1             |
| 3     | A00.4              | F   | 3181.274579       | 3682.151115       | NaN               | 3181.274579          | 3682.151115       | 2             |
| 4     | A00.5              | F   | 4042.404631       | 4042.404631       | NaN               | 4042.404631          | 4042.404631       | 1             |
| ...   | ...                | ... | ...               | ...               | ...               | ...                  | ...               | ...           |
| 20945 | Z99.3              | F   | 2975.364418       | 5655.832483       | NaN               | 2975.364418          | 5655.832483       | 2             |
| 20946 | Z99.3              | M   | 4087.810795       | 4230.383307       | NaN               | 4087.810795          | 4230.383307       | 2             |
| 20947 | Z99.7              | F   | 3805.511272       | 4296.366234       | NaN               | 3805.511272          | 4296.366234       | 2             |
| 20948 | Z99.8              | F   | 3767.443903       | 3767.443903       | NaN               | 3767.443903          | 3767.443903       | 1             |
| 20949 | Z99.9              | F   | 4975.695772       | 4975.695772       | NaN               | 4975.695772          | 4975.695772       | 1             |

20950 rows x 8 columns

## 4 Part 4: Strategic Insights and Recommendations

### 4.1 Charge Discrepancies Among AR-DRG Codes

The top 10 DRGs accruing the largest total charges include DRG002, DRG003, and DRG001, as shown in the bar chart. These DRGs significantly drive the hospital's total charges, indicating that procedures under these groups are likely more resource-intensive. By closely monitoring and optimizing resource utilization for these DRGs, Ramsay can improve operational efficiency. For example, understanding the specific cost drivers (e.g., theatre time, prosthesis, bundled services) could lead to more targeted cost-saving initiatives, such as negotiating better supplier contracts for high-cost items or improving scheduling efficiency in high-cost departments like the operating theatres.

### 4.2 Variability in TotalCharges by PrincipalDiagnosis

The analysis of TotalCharges by PrincipalDiagnosis and Sex reveals variability in cost distribution across different diagnoses. Some diagnoses, such as Z99.3 (diagnoses with the highest charges), exhibit wide charge variability. Identifying why some cases incur much higher charges can help Ramsay address inefficiencies or inconsistencies in treatment protocols. This could also be an opportunity to improve patient care by streamlining treatment pathways for certain conditions, ensuring that costs are aligned with the expected outcomes, and reducing unnecessary procedures or interventions.

## 5 Part 5: Model Development

In this analysis, we aimed to predict the total hospital charges (TotalCharges) based on a variety of patient features. Specifically, the dataset comprised both numeric and categorical features. Numeric features included Age and LengthOfStay, which capture key aspects of a patient's stay. Meanwhile, the categorical features encompassed a range of information including AR-DRG (Diagnosis-Related Group), Principal\_ProcedureCode, CareType, SourceOfReferral, UrgencyOfAdmission, AgeGroup, WeekendStay, and Sex, which provide context about the medical procedures and patient demographics.

To ensure that the data was suitable for machine learning models, the dataset was split into training and testing sets, with 70% of the data used for training and 30% held out for testing. For preprocessing, a pipeline was implemented: numeric features underwent median imputation for missing values, followed by standard scaling to normalize the data. Categorical features were processed using the most-frequent imputation and transformed into numerical form through one-hot encoding.

We evaluated four different machine learning models—RandomForest, LinearRegression, Support Vector Regression (SVR), and GradientBoosting—by integrating the preprocessing pipeline with each model to streamline training and prediction. After fitting the models to the training data, they were saved to disk for future deployment in the prediction pipeline.

Model performance was assessed using several metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), R<sup>2</sup> Score, and Mean Percentage Error (MPE). The table below (Table ??) shows the performance of each model across these metrics.

**Table 1:** Performance Metrics for Different Regression Models

| Model            | MAE    | RMSE    | R <sup>2</sup> Score | MPE (%) |
|------------------|--------|---------|----------------------|---------|
| RandomForest     | 647.45 | 929.82  | 0.13                 | 0.17    |
| LinearRegression | 846.62 | 1190.18 | -0.43                | 0.22    |
| SVR              | 735.26 | 997.00  | -0.00                | 0.20    |
| GradientBoosting | 729.01 | 983.63  | 0.03                 | 0.19    |

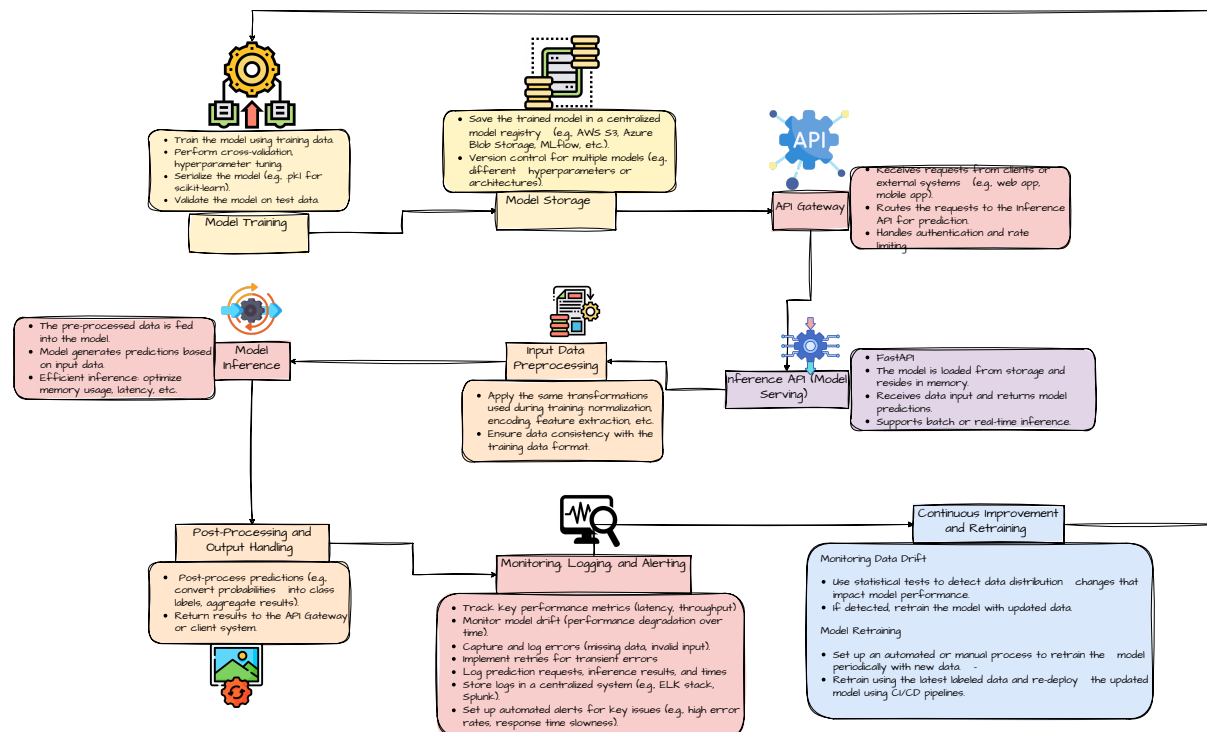
As indicated by the results, the RandomForest model achieved the best overall performance, with the lowest MAE (647.45) and RMSE (929.82) values, as well as the highest R<sup>2</sup> score (0.13), indicating that it captured the most variance in the data compared to the other models. The GradientBoosting model also showed



competitive performance, with an MAE of 729.01 and an RMSE of 983.63, and a slight improvement in the  $R^2$  score over SVR and LinearRegression.

LinearRegression performed the worst, with a negative  $R^2$  score (-0.43), indicating that the model was not able to effectively capture the variance in the data. SVR performed moderately, but its  $R^2$  score was close to zero, showing it struggled to generalize well. In summary, the RandomForest model emerges as the most promising option for predicting total hospital charges, providing more accurate and consistent results across the metrics.

## 6 Part 6: MLOps and Deployment



**Figure 4:** Architecture for model deployment

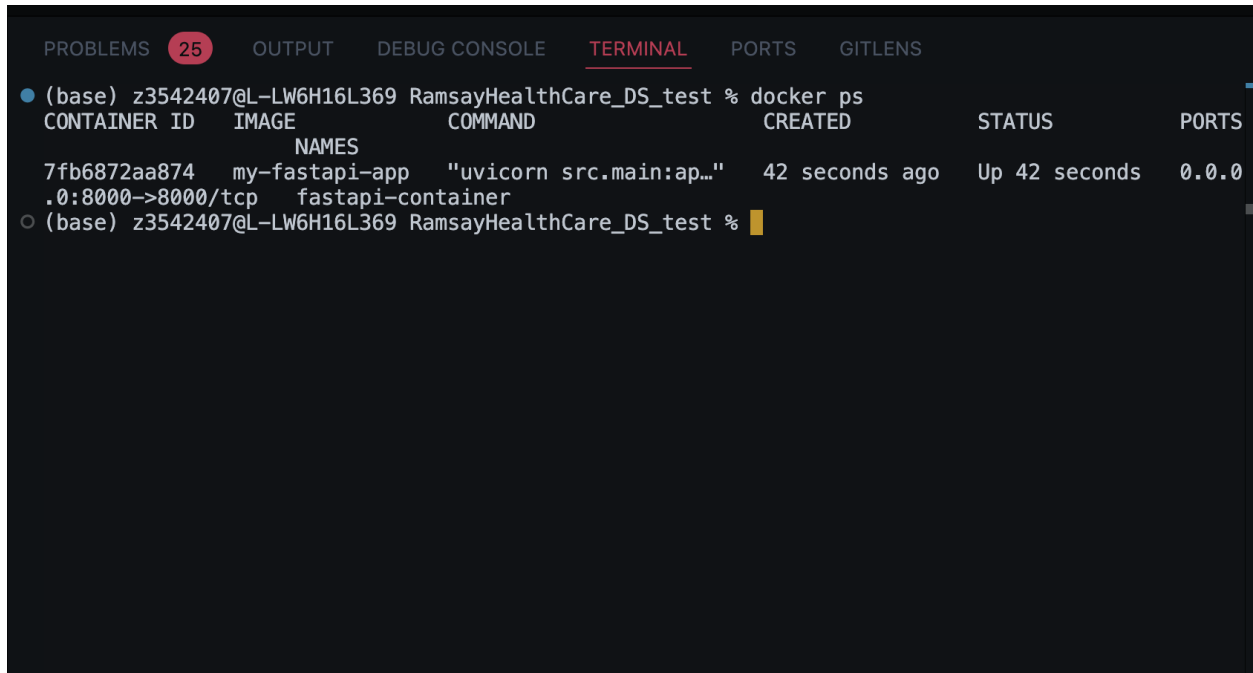
The figure 6 demonstrates the architecture for deploying the model into a production environment. The deployment process begins with model training, where techniques such as hyperparameter tuning and validation are applied to optimize performance. Once the model is trained, it is serialized and securely stored in a centralized cloud storage solution (such as AWS S3 or Azure Blob), enabling version control to track different model architectures or hyperparameter configurations.

For inference, the input data undergoes the same preprocessing steps that were applied during the training phase, ensuring consistency in feature transformation. An API gateway handles incoming requests from external clients (such as web or mobile applications) and routes them to the inference API. This API loads the model from storage and generates predictions in real-time, with optimizations in place to minimize memory consumption and reduce latency for quick responses.

Post-processing is then applied to the predictions, ensuring the results are properly formatted and sent back to the API gateway or directly to the client. The system also incorporates robust monitoring and logging mechanisms to track performance metrics such as latency and throughput, detect potential model drift, and manage any errors. Centralized log storage and automated alerts ensure that issues can be identified and resolved proactively.

Continuous monitoring of data drift is essential for identifying shifts in data distributions that could impact model accuracy. When such drift is detected, the model is retrained using the latest data and seamlessly

redeployed through CI/CD pipelines, ensuring that the model stays up to date and continues to deliver accurate predictions over time.



```
(base) z3542407@L-LW6H16L369 RamsayHealthCare_DS_test % docker ps
```

| CONTAINER ID      | IMAGE             | COMMAND                  | CREATED        | STATUS        | PORTS |
|-------------------|-------------------|--------------------------|----------------|---------------|-------|
| 7fb6872aa874      | my-fastapi-app    | "uvicorn src.main:ap..." | 42 seconds ago | Up 42 seconds | 0.0.0 |
| .0:8000->8000/tcp | fastapi-container |                          |                |               |       |

```
(base) z3542407@L-LW6H16L369 RamsayHealthCare_DS_test %
```

Figure 5: Run docker

default

GET / Read Root

POST /predict Predict

Cancel

Reset

Parameters

No parameters

Request body required

application/json

```
{
  "Age": 62,
  "LengthOfStay": 18,
  "AR_DRG": "DRG002",
  "Principal_ProcedureCode": "16071-02",
  "CareType": "Emergency",
  "SourceOfReferral": "Specialist",
  "UrgencyOfAdmission": "Urgent",
  "AgeGroup": "Adult",
  "WeekendStay": 1,
  "Sex": "M"
}
```

Servers

These operation-level options override the global server options.

/

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8000/predict' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "Age": 62,
    "LengthOfStay": 18,
    "AR_DRG": "DRG002",
    "Principal_ProcedureCode": "16071-02",
    "CareType": "Emergency",
    "SourceOfReferral": "Specialist",
    "UrgencyOfAdmission": "Urgent",
    "AgeGroup": "Adult",
    "WeekendStay": 1,
    "Sex": "M"
  }'
```

Request URL

http://localhost:8000/predict

Server response

| Code | Details   |
|------|---|
| 200  | <div>Response body</div> <div><pre>{   "prediction": 4287.052842430948 }</pre></div> <div><div>Download</div></div> |

Response headers

Figure 6: FastAPI