# Heart Attack Risk Model: Results

1. **Quantitative Outcomes**:
    - Summary table comparing linear model performance:

| Model | Accuracy | Precision | Recall | F1 Score | AUC-ROC |
|---|---|---|---|---|---|
| Logistic Regression | 0.90 | 0.93 | 0.88 | 0.90 | 0.94 |
| Support Vector Machines | 0.90 (after hyperpara-meter tuning) | 0.90 | 0.84 | 0.87 | 0.87 |
| Decision Tree | 0.79 | 0.85 | 0.72 | 0.78 | 0.79 |
| Random Forest | 0.79 | 0.85 | 0.72 | 0.78 | 0.79 |
| Gradient Booster | 0.87 | 0.90 | 0.84 | 0.87 | 0.87 |

    - Feature importance rankings from the models: Since the two best models were SVM and Logistic Regression, we will examine the coefficients of the features to determine the most important ones. We will use an absolute value of 0.5 as the minimum required to be classified as important:

        i.

| Support Vector Machines | |
|---|---|
| **Feature** | **Coefficient** |
| previous_peak | -0.51 |
| sex | -0.60 |

| Support Vector Machines | |
|---|---|
| exercise_induced_angina_1 | -0.61 |
| num_major_vessels_1 | -1.33 |
| num_major_vessels_2 | -1.64 |
| num_major_vessels_3 | -1.00 |
| num_major_vessels_4 | 1.00 |
| chest_pain_type_2 | 0.96 |
| chest_pain_type_3 | 1.14 |
| thal_rate_2 | 0.62 |
| thal_rate_3 | -0.52 |

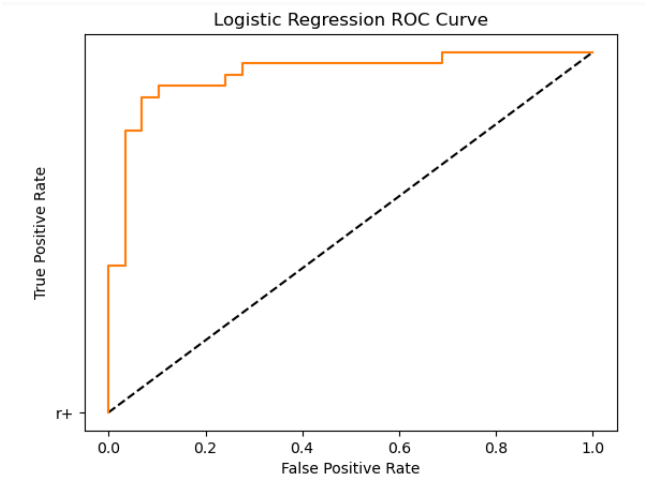
ii.

| Logistic Regression | |
|---|---|
| **Feature** | **Coefficient** |
| previous_peak | -0.78 |
| sex | -1.17 |
| exercise_induced_angina_1 | -0.85 |
| num_major_vessels_1 | -1.68 |
| num_major_vessels_2 | -1.86 |
| num_major_vessels_3 | -1.07 |
| chest_pain_type_2 | 1.31 |
| chest_pain_type_3 | 1.34 |

| Logistic Regression | |
|---|---|
| slope_2 | 0.62 |
| thal_rate_2 | 0.53 |
| thal_rate_3 | -0.77 |
| **INTERCEPT:** | **1.15** |

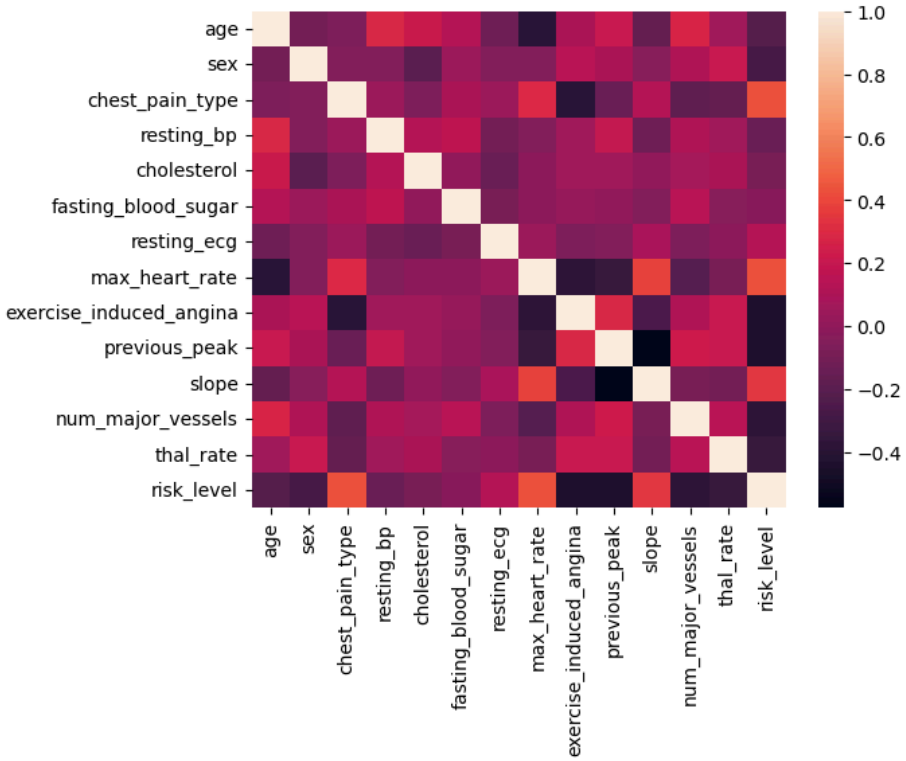
**Observations on the above models:** Logistic Regression performed best in all metrics. In both models there were 11 features that were significant in predicting the target variable, exactly half of the 22 features in consideration. SVM included **num_major_vessels_4**, while Logistic Regression did not; and LR included **slope_2**, while SVM did not. Otherwise, the same features were significant in both models. LR predicts a 1.15 intercept, which is of some interest since this number is larger than 1, meaning that when the features are all negligible the model predicts a heart-attack above the max value of 1.

2. **Visual Insights**:
    - ROC curve from Logistic Regression:



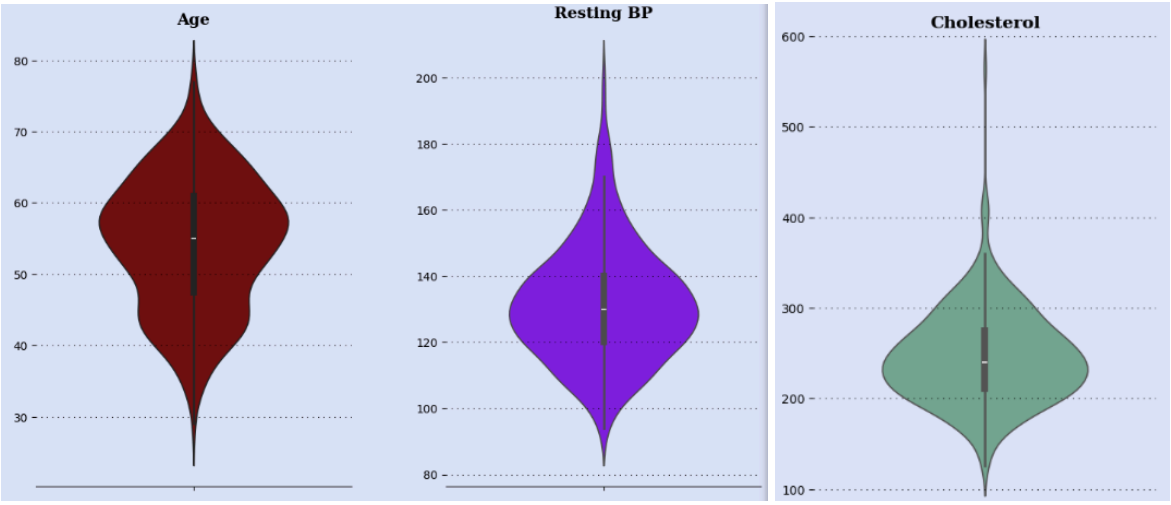

**Observation:** The ROC has a significant amount of area under it and above the diagonal.This illustrates the usefulness of the model.

    - Heatmap of all features and target:



**Observations from the above heatmap:** If we look across the bottom row, we observe the relatively high correlations between the target, **risk_level,** and the features that turned out to be significant in the models: **sex, chest_pain_type, exercise_induced_angina, previous_peak, num_major_vessels,** and **thal_rate**.

3. **Detailed Performance Analysis**:
    - Clearly the Logistic Regression model outperformed all the others, tying only with respect to accuracy with SVM. For features significant in both models, the coefficients agree on polarity. However, while in both models **num_major_vessels_1,2 and 3** have negative coefficients, **num_major_vessels_4** has a positive coefficient in SVM.
    - While most of the significant features are ones which would be expected, based on widely-published and decades-old research, the absence of features like age, resting blood pressure, cholesterol, and resting ECG could be viewed as surprising. This could, however, be explained by the relatively small sample size of this study. Also, as seen here, some of these features are fairly normally distributed, except for a few outliers:




4. **Recommendations and Limitations**:
    - Physicians and healthcare providers can use this model to identify high-risk patients and prioritize preventative interventions.
    - Dataset size (n=303) limits generalizability.
    - Further validation needed on diverse populations.

5. **Supporting Documents**:
    - Reference key visualizations or detailed findings included in the final report.

The "Results" section is detailed and geared toward those who need an in-depth understanding of the outcomes, such as researchers, healthcare practitioners, or collaborators.

The ReadMe, on the other hand, serves as a **project overview and entry point** for anyone visiting your repository. It should include:

1. **Project Objective**:
    - A concise summary of the problem you're solving and its importance.
2. **High-Level Results**:
    - A quick snapshot of the model's performance (e.g., "Gradient Boosted Trees achieved an accuracy of 88% with an AUC-ROC of 0.91").
    - Mention top predictors like "Previous Peak and Number of Major Vessels."
3. **Key Features of the Repository**:
    - Highlight sections like "Code," "Data Preparation," "Model Training," and "Results."
4. **Usage Instructions**:
    - Steps for setting up the environment and running the code.
    - Dependencies and installation instructions.
5. **Link to Detailed Results**:
    - Direct users to the "Results" folder or section for a comprehensive breakdown.
6. **Interactive or Public Resources**:
    - Include links to interactive notebooks, a hosted dashboard, or documentation.

The ReadMe is for a broader audience and serves as a gateway to your project, enticing readers to dive deeper.

If you align the "Results" section and the ReadMe with these purposes, they won't overlap excessively but will reinforce each other effectively.

## Here are a few suggestions for your repository:

**ReadMe.md**

Expand your ReadMe to include:

1. **Overview**:
    - Briefly describe the project and its purpose: predicting heart attack risk using machine learning.
2. **Dataset**:
    - Provide a description of the dataset, link to it, and highlight key features (e.g., size, variables).
3. **Steps to Reproduce**:
    - Instructions for setting up the environment and running the notebooks.

    - Include dependencies and how to install them (e.g., via `requirements.txt` or `environment.yml`).
4. **Key Results**:
    - Summarize the performance of your best model (e.g., "Achieved 88% accuracy using Gradient Boosted Trees").
    - Highlight any significant findings (e.g., "Previous Peak is the most influential predictor").
5. **Repository Structure**:

List the purpose of each folder, e.g.:

```
- Data: Contains the CSV dataset used in this project.
- Notebooks: Jupyter notebooks for data preprocessing, modeling, and evaluation.
- Results: Outputs of the models and visualizations.
- Exploratory Data Analysis: Scripts and plots from the EDA phase.
```

6. **Future Work**:
    - Suggestions for expanding the analysis or model improvements.

**Results Folder**

Include:

1. **Key Visualizations**:
    - Confusion matrices, AUC-ROC curves, feature importance plots.
    - Heatmaps or other visual summaries from your EDA.
2. **Performance Metrics**:
    - Tables or charts comparing model performance (e.g., logistic regression vs. Gradient Boosted Trees).
3. **Interpretability Tools**:
    - SHAP/feature importance analysis results showing which features contribute most to predictions.
4. **File Format**:
    - Use PDFs or Markdown files to summarize findings for easy readability.

## Current File Organization

**Folders:**

1. **Combine Similar Content**:
    - Merge `.ipynb_checkpoints` into `Notebooks` for clarity.

- Ensure that each folder serves a unique purpose.

2. **Data Folder**:
   - Add a `README.md` inside the folder to describe the dataset (e.g., source, purpose, and preprocessing) - pretty much what you have in the README.md file currently, but just with a little more info.
3. **Exploratory Data Analysis**:
   - Provide summary insights (currently detailed in your PDF) in a Markdown file for easier reference.
4. **Results**:
   - Ensure this includes both raw outputs and summary visualizations for models.

**File Suggestions:**

1. **Requirements File**:
   - Include `requirements.txt` or `environment.yml` to specify dependencies.
2. **Automated Workflow**:
   - Add `.github/workflows` for CI/CD if you plan to automate tests or deployments in the future (or allow that capability)