Hybrid Data Science Methodology Titanic

Rex Coleman

2024-11-13

Hybrid Data-Science Methodology Titanic

Executive Summary

Welcome to my Framework for Solving Data Science Problems. This repository provides a comprehensive framework for solving data science problems. The project builds upon one of the most popular Kaggle notebooks, leveraging best-in-class methodologies to create a reliable foundation for solving data science problems. By reproducing and substantially building upon this work, I aim to illustrate the value of learning from top practitioners while also solving one of the most important problems in data science.

Rushing into a data science project without a structured approach can lead to numerous problems, which can severely impact project success, cost, and outcomes. This project addresses these issues by implementing a well-defined framework and best practices ensuring thorough problem understanding, effective data preprocessing, and robust model evaluation.

Borrowing fron the giants of Agile, DevOps, and Lean Entrepreneurship, we are leveragine the concept of 'shifting left' to support a more flexible and adaptive development process, facilitating faster delivery of high-quality data science solutions that originate from clearly defined business needs.

This project applies the above concepts to the popular "Titanic - Machine Learning from Disaster" Kaggle competition and can applied generally to a wide array of data science problems.

Thank you for visiting my repository. I hope this project inspires you to implement a structured approach to avoid common data science pitfalls. I welcome comments: especially those that will help improve upon this concept.

Figure 1: Model Accuracy - This plot shows train, validate and test model accuracies in Kaggle test accuracy order. The top four models (BaggineClassifier, BernoulliNB, XGBClassifier and EnsembleHardVoting) outperformed both hard and soft voting ensemble models. The Baseline Handmade Decision Tree model several other models.

Table of Contents

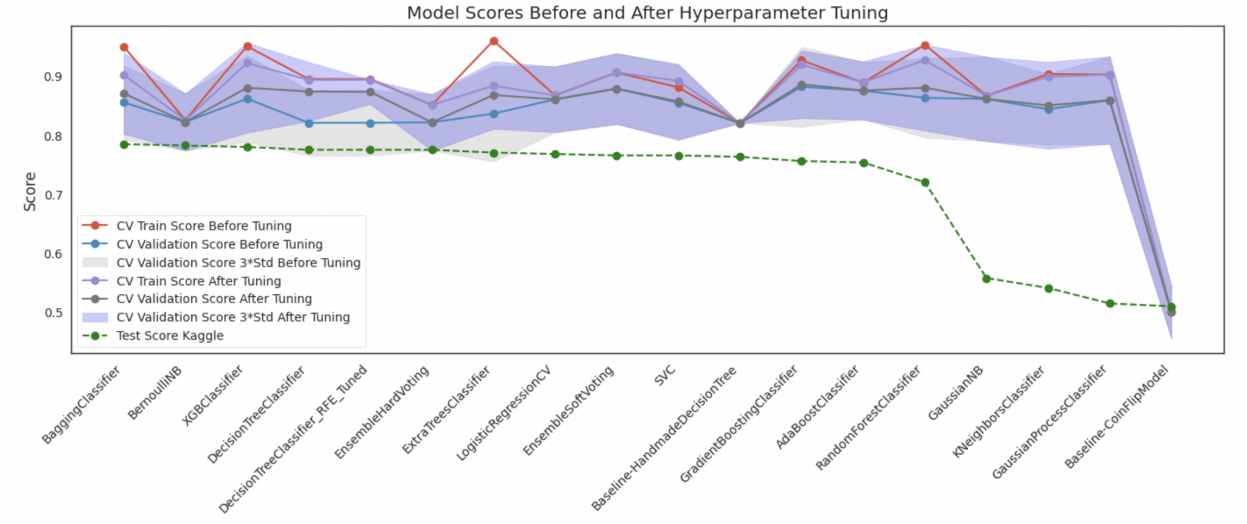

Figure 1: Model results table

8. References
9. Contact

---

1. Introduction

The motivation behind this project is twofold:

Learning from the Best: By reproducing work from top data scientists, we can gain valuable insights and understand the methodologies that lead to successful projects. **Framework Approach**: Creating a robust framework for data science projects that can be applied to various datasets and problems, ensuring a structured approach to avoid common pitfalls.

1.1 Learning from the Best

Why reinvent the wheel when you don't have to. To mix metaphors, let's stand on the shoulders of giants and improve upon their work!

1.2 Framework Approach

1.2.1 Problems From Ruching Into a Data Science Project Rushing into a data science project without a structured approach can lead to numerous problems, severely impacting the success, reliability, and cost of the outcomes. Below is an exhaustive list of these potential issues:

Inadequate Problem Understanding: - Misalignment with business objectives. - Unclear problem definition leading to irrelevant solutions.

Poor Data Collection and Exploration: - Missing critical data points. - Overlooking data quality issues. - Failure to understand the data distribution and patterns.

Insufficient Data Cleaning and Preprocessing: - Presence of noisy or irrelevant data. - Incorrect handling of missing values. - Inconsistent data formatting and scaling.

Ineffective Feature Engineering: - Missing out on key features that improve model performance. - Overfitting due to too many features. - Ignoring domain knowledge in feature selection.

Inappropriate Model Selection and Training: - Choosing models that are not suitable for the problem. - Not validating the model selection process. - Inadequate training leading to underfitting or overfitting.

Lack of Proper Model Evaluation and Validation: - Using incorrect metrics for model evaluation. - Not performing cross-validation to ensure model generalization. - Ignoring potential data leakage during validation.

Inadequate Hyperparameter Tuning: - Suboptimal model performance due to default hyperparameters. - Time-consuming trial and error without a systematic approach.

Misinterpretation of Results: - Drawing incorrect conclusions from model outputs. - Failure to consider model limitations and biases.

Poor Model Deployment (if applicable): - Incompatibility with production environment. - Lack of monitoring and maintenance plan.

Inadequate Documentation and Reporting: - Difficult for others to understand and reproduce the work. - Lack of transparency in methodologies and results.

By addressing these issues through a structured approach, as demonstrated in this project, we can significantly improve the quality and reliability of data science outcomes.

1.2.2 Benefits of Leveraging a Data Science Framework:

1. **Enhanced Quality and Reliability**: Early identification and resolution of issues improve the overall reliability of the project.
2. **Cost and Time Efficiency**: Addressing issues early reduces the cost and time associated with fixing problems later.
3. **Better Stakeholder Engagement**: Early involvement of stakeholders ensures the project remains aligned with business goals.
4. **Structured Approach**: A methodical approach from the beginning ensures a systematic way to tackle data science projects.

This framework serves as a guide for tackling data science projects methodically and effectively.

---

2. Data Science Framework

2.1 Define the Problem

If data science, big data, machine learning, predictive analytics, business intelligence, or any other buzzword is the solution, then what is the problem? Problems should precede requirements, requirements should precede solutions, solutions should precede design, and design should precede technology.

Kaggle clearly defined this problem for us.

2.2 Gather the Data

Chances are, the dataset(s) already exist somewhere. It may be external or internal, structured or unstructured, static or streamed, objective or subjective. The goal is to find and consolidate these datasets.

Kaggle provided a clean dataset.

2.3 Prepare Data for Consumption

Data wrangling is a required process to turn "wild" data into "manageable" data. This includes data extraction, data cleaning, and preparing data for analysis by implementing data architectures, developing data governance standards, and ensuring data quality.

This project uses the 4 C's of Data Cleaning: Correcting, Completing, Creating, Converting.

2.4 Perform Exploratory Data Analysis

Deploy descriptive and graphical statistics to look for potential problems, patterns, classifications, correlations, and comparisons in the dataset. Data categorization is also important to select the correct hypothesis test or data model.

2.5 Model Data

Data modeling can either summarize the data or predict future outcomes. The dataset and expected results determine the algorithms available for use. Algorithms are tools that must be selected appropriately for the job.

This project includes a generalized framework for model selection. This report presents a baseline back-of-the-envelope calculated decision tree and compares resultes from multiple models. This report also implementes the following techniques to improve model results: ensembling, hyper-parameter tuning and recursive feature elimination.

2.6 Validate and Implement Data Model

Test your model to ensure it hasn't overfit or underfit your dataset. Determine if your model generalizes well by validating it with a subset of data not used during training.

Cross validation was used to improve generalization.

2.7 Optimize and Strategize

Iterate through the process to make the model better. As a data scientist, your strategy should be to focus on recommendations and design while outsourcing developer operations and application plumbing.

---

3. Project Details

- **Dataset**: Kaggle Titanic Dataset
- **Objective**: The competition objective is simple: use machine learning to create a model that predicts which passengers survived the Titanic shipwreck.

Project Details

Reproducing Best-in-Class Work

Reproducing high-quality work from leading data scientists provides several benefits: - **Benchmarking**: Establishes a performance benchmark to compare our models against. - **Learning**: Understand the best practices and methodologies used by top practitioners. - **Innovation**: Builds a foundation upon which new ideas and improvements can be developed.

Technologies Used

- **Programming Languages**: Python version 3.10.13
- **Libraries**: Pandas, NumPy, Matplotlib, Seaborn, SciPy, Scikit-learn, itertools, Graphviz, os, sys, IPython, random, time, XGBoost
- **Tools**: Jupyter Notebooks, Git, GitHub, Kaggle

Getting Started

Follow these instructions to get a copy of the project up and running on your local machine.

1. **Clone the repository**: `bash git clone git@github.com:rexcoleman/GeneralizedDataScienceFramework-T`
2. **Install dependencies**: `bash pip install -r requirements.txt`
3. **Run the Jupyter Notebook**: `bash jupyter notebook`

Results and Insights

The project results include detailed analysis, model performance metrics, and visualizations that provide insights into the predictive power of the models used.

Figure 1: Model Accuracy - This plot shows train, validate and test model accuracies in Kaggle test accuracy order. The top four models (BaggineClassifier, BernoulliNB, XGBClassifier and EnsembleHardVoting) outperformed both hard and soft voting ensemble models. The Baseline Handmade Decision Tree model several other models.

Figure 2: Model Accuracy Table - This table shows train, validate and test model accuracies in descending Kaggle test accuracy order. The top four models (BaggineClassifier, BernoulliNB, XGBClassifier and EnsembleHardVoting) outperformed both hard and soft voting ensemble models. The Baseline Handmade Decision Tree model several other models.

Figure 3: Model Error Plot - This plot compares model error across multiple models. Bias error is defined as perfect accuracy minus train accuracy. Variance is defined as test error and train error. Typically it is better to use the difference in dev errer (validation error) and training error. In the case of our models, there is a wide margin between test error and validation error so I am including it in my variance error calculation.

As a general rule for model performance, we want to work on improving the greater error (bias or variance).

We can potentially improve model bias with: - Use a larger neural network - Train longer - Use better optimization algorithms, e.g. momemtum, RMSProp, Adam - Search for better architecture/hyperparameters, e.g. RNN, CNN

We can potentially improve model variance with: - More data - Regularization, e.g. LS, droppout, data augmentation - Search for better architecture/hyperparameters, e.g. RNN, CNN

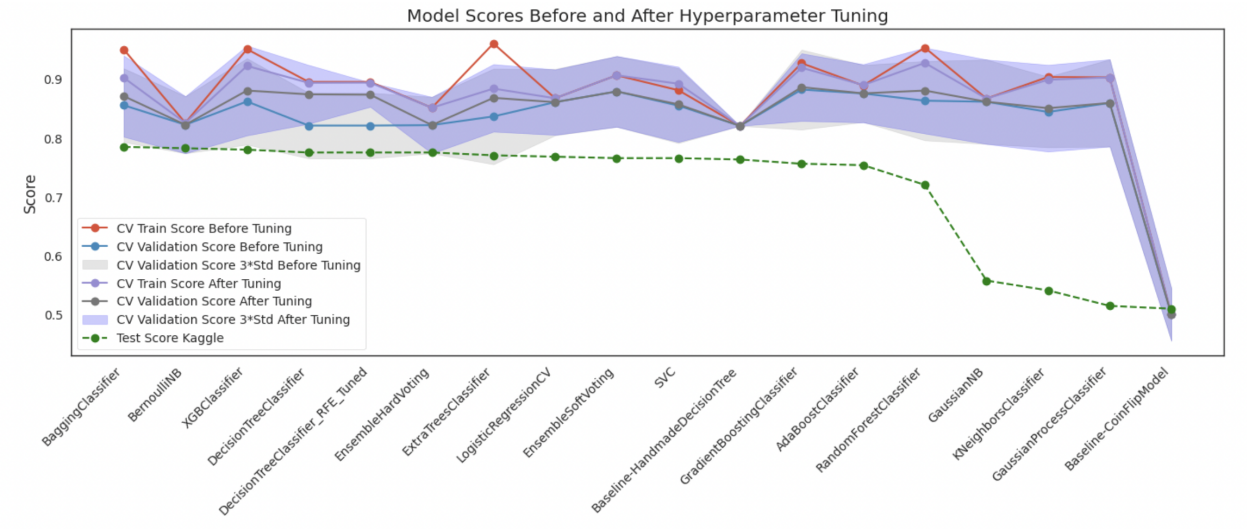

Figure 2: Model results table

Model	CV Train Score Before Tuning	CV Validation Score Before Tuning	CV Validation Score 3*Std Before Tuning	CV Train Score After Tuning	CV Validation Score After Tuning	CV Validation Score 3*Std After Tuning	Test Score Kaggle	Bias Indicator	Variance Indicator
BaggingClassifier	0.950370	0.855483	0.062583	0.902254	0.870918	0.069051	0.78468	0.079452	0.086238
BernoulliNB	0.825397	0.822248	0.048543	0.825397	0.822248	0.048543	0.78229	0.003148	0.039958
XGBClassifier	0.950756	0.861856	0.073333	0.922306	0.880438	0.076357	0.77990	0.070318	0.100538
DecisionTreeClassifier_RFE_Tuned	0.895131	0.820896	0.055746	0.893892	0.873408	0.020699	0.77511	0.021723	0.098298
DecisionTreeClassifier	0.895131	0.820896	0.055746	0.893529	0.873965	0.049985	0.77511	0.021166	0.098855
EnsembleHardVoting	0.851124	0.821642	0.047965	0.851124	0.821642	0.047965	0.77511	0.029482	0.046532
ExtraTreesClassifier	0.960401	0.836407	0.081272	0.883710	0.867991	0.057349	0.77033	0.092410	0.097661
LogisticRegressionCV	0.867317	0.860694	0.056413	0.867306	0.860660	0.055554	0.76794	0.006657	0.092720
SVC	0.881016	0.854789	0.063689	0.891980	0.856945	0.063946	0.76555	0.024070	0.091395
EnsembleSoftVoting	0.906373	0.878927	0.060138	0.906373	0.878927	0.060138	0.76555	0.027446	0.113377
Baseline-HandmadeDecisionTree	0.820426	0.820426	0.000000	0.820426	0.820426	0.000000	0.76315	0.000000	0.057276
GradientBoostingClassifier	0.926910	0.882039	0.067880	0.919254	0.886196	0.057457	0.75598	0.040714	0.130216
AdaBoostClassifier	0.888960	0.875442	0.048490	0.889708	0.875656	0.049489	0.75358	0.013305	0.122076
RandomForestClassifier	0.953326	0.863178	0.067368	0.927422	0.880434	0.072966	0.72009	0.072892	0.160344
GaussianNB	0.866791	0.861380	0.071747	0.866791	0.861380	0.071747	0.55741	0.005410	0.303970
KNeighborsClassifier	0.903513	0.843984	0.060027	0.898847	0.850436	0.073728	0.54066	0.053077	0.309776
GaussianProcessClassifier	0.902859	0.859381	0.074157	0.902859	0.859381	0.074157	0.51435	0.043478	0.345031
Baseline-CoinFlipModel	0.499450	0.499450	0.044690	0.499450	0.499450	0.044690	0.50956	0.000000	-0.010110

Figure 3: Model results plot

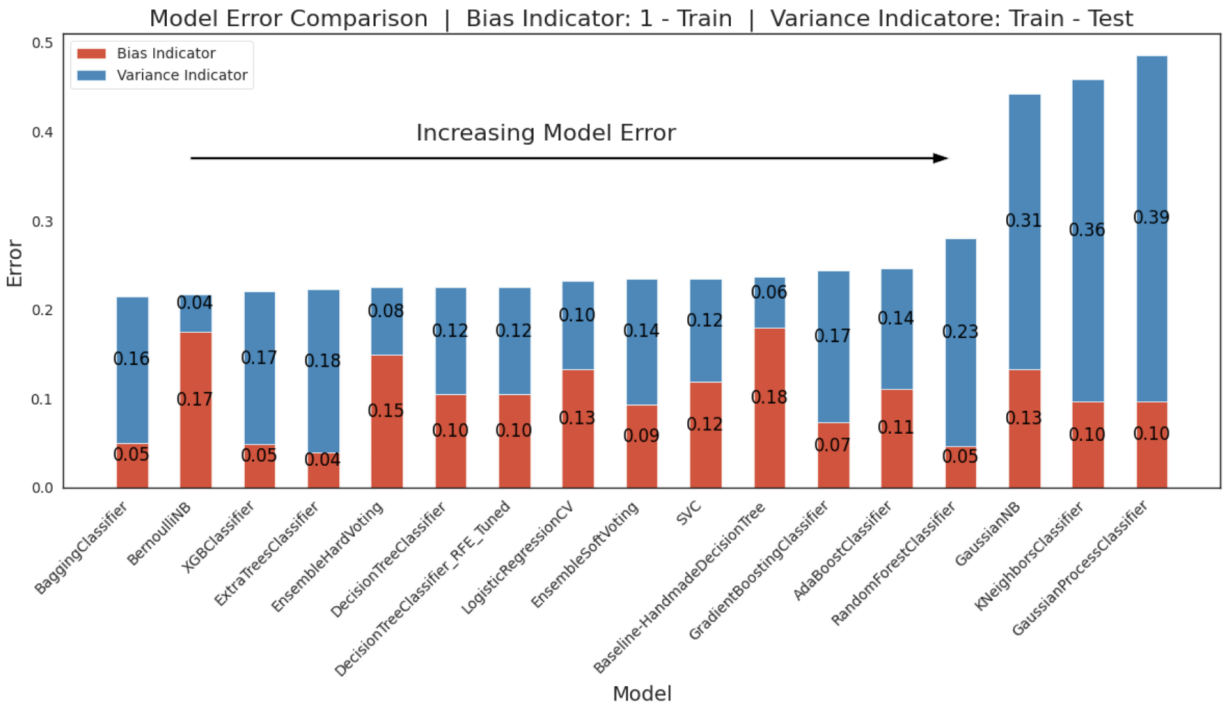

Figure 4: Model variance indicator

Observations:

1. The Bagging Classifier model produced the highest Kaggle accuracy score: **0.78468**.
2. The Bagging Classifier model performed better than the ensemble model Kaggle accuracy scores: **0.77511** (hard voting), **0.76555** (soft voting).
3. The Bagging Classifier model appears to have the lowest variance compared to the other models.
4. The Bagging Classifier model underperformed when compared to the TensorFlow Decision Forest model with a Kaggle accuracy score of **0.80143**.

Areas For Future Research:

1. Why does the TensorFlow Decision Forest model outperform all models in this notebook?
2. Why do several models in this notebook outperform the ensemble models? How can we improve the ensemble models? When does ensembling shine?
3. What hyperparameters tuning can I use to improve performance?
4. How are some of the submissions achieving 100% accuracy Kaggle scores?
5. Find coorelation heat map code that doesn't require forcing Kaggle version control.

---

6. References

- **Original Kaggle Notebok**:
  - Freeman, 2016, A Data Science Framework: To Achieve 99% Accuracy
- **Original Kaggle Notebok**:

– Titanic - Machine Learning from Disaster