# LLM Feature Selection makes Best Subset Selection feasible on high dimensional datasets

By: Kian Drees - Advisor: Dr. Weijun Xie

## 1. Introduction

In this paper, I discuss my framework for using large language models to assist in feature selection for Best Subset Selection models. (regression and SVM) This differs significantly from previous work on LLM lasso as it allows the LLM feature selection process to be used with best subset selection, rather than lasso, making for more interpretable and explainable models, as the exact number of features and which features are used is easily obtainable.

This framework is best applied when there is high number of features that would be difficult or time consuming to analyze their importance and feature select manually; the LLM does the job of narrowing the feature pool using domain specific knowledge in the form of context given by the user, and then passes the smaller pool of features onto the BSS, to find the optimal subset of. In this way, high feature datasets can easily and quickly be analyzed and predictions made, without the data scientist needing to have an intimate understanding of the topic of the data and with the highest importance features already selected.

LLM BSS improves upon normal best subset selection by allowing it to perform efficiently on high dimensional datasets that would normally be infeasible or suboptimal to use best subset selection on. LLM feature selection allows the model to make insights through details about the subject matter rather than focusing on the numbers alone, and most importantly narrows the search space of the BSS to a manageable amount while taking only the features the LLM deems to be most applicable/useful.

In doing this, LLMBSS unlike LLM Lasso is able to reliably create a sparse model that avoids the curse of dimensionality; that is, it avoids overfitting to the noise and is still able to identify overarching patterns in the data. Below is shown general rule of thumb, that past a certain point, additional features added to a machine learning model actually decrease it's effectiveness due to the reasons stated previously. This is why precise feature selection is so important in machine learning, and what makes LLMBSS so powerful.

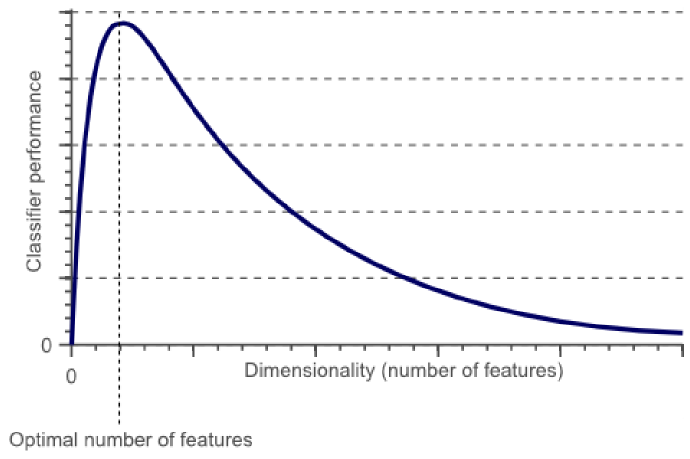


*Figure 1: Additional features only make a model stronger up to a certain point. Past that, the model overfits and performs worse.*

In addition, a lower feature amount just makes computation faster and cheaper, which, if the optimizer used has a time limit set, could even affect performance.

**2. Regression**

I began my research by experimenting with LLM feature selection for regression best subset selection with Gurobi, which works by minimizing the Residual Sum of Squares , represented in this formula,

$$\text{Min} \quad Z = \beta^T X^T X\beta - 2y^T X\beta + y^T y$$

derived in this way;

$$e^T e = (y - X\beta)^T (y - X\beta) = \beta^T X^T X\beta - 2y^T X\beta + y^T y$$

The BSS model additionally imposes a budget constraint to limit the number of nonzero features to be less than or equal to a specified amount k, through L0 norm regularization with the following constraint:

$$||\boldsymbol{\beta}'||_0 \leq k$$

The Gurobi machine then optimizes this problem by iteratively changing the intercept and weights on each feature. These weights and the intercept make up the trained model; to calculate the prediction vector, all you need to do is right multiply the X testing matrix by the weight/coefficient vector and add the intercept to each scalar.

My LLM framework works in two main steps. First, it sends a call to the LLM, including context for the problem, a specific prewritten line of instructions on the task and how to format the response, and the specific features from which to choose from. The layout for this message is shown in table 9, with the exact instructions used. Context for the subject was created only with information available on the page for the dataset. The LLM returns a sequence of features ranked in their order of importance, which is then sliced to narrow down which features are passed to the optimizer.

The second step is to use Gurobi to solve and optimize for the problem. This is done using the formulas detailed above, in the form of Gurobi variables which are tweaked during optimization to find the best coefficients for each feature, Gurobi constraints defining the constraints of the problem, and the Gurobi objective, defining the equation that is to be minimized. Finally, the problem is optimized, and the best weights and intercept found in the time limit are returned.

I begin testing the framework with a small dataset about predicting Spotify streams, with 31 features after data preprocessing. For this dataset, the BSS model looks through all the features and selects 10 or less to be used in the final predictions, while the LLM model uses the LLM (in this case Gemini) to select the top 20 features from the original 31, then uses BSS to select 10 or less. The Rand model picks 20 features at random and then uses BSS to select 10 or less. Unless otherwise specified, the data is split into a 20% test 80% train split.

As can be seen in (table 2), the LLM model performs comparably to the BSS one, both at an average of about .74 r^2 score over 10 trials, while the Rand model falls behind, at an average of around .66 r^2, also over 10 trials. The performance of these models is also displayed visually in these graphs relating r^2 and mse for each model at every seed:

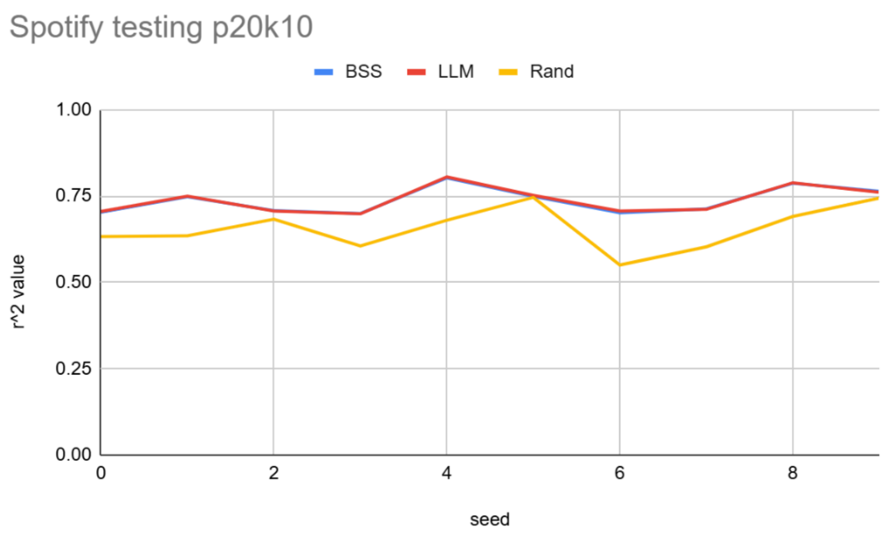


*Figure 2: Spotify r^2 value for 3 models on 10 different trials, each using different seeds for train and test split.*

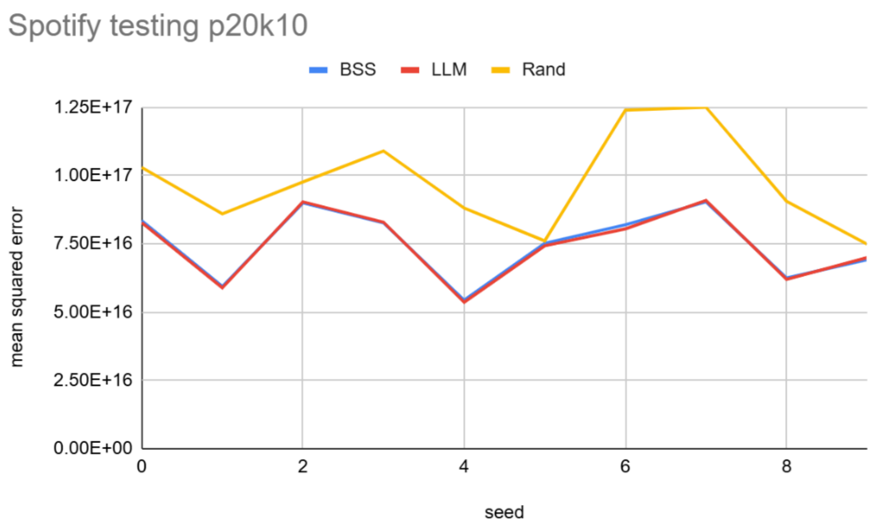


*Figure 3: Spotify mean squared error for 3 models on 10 different seeds.*

Because this is a small feature dataset, the BSS model takes a short amount of time to train (table 1), an average of about 0.25 seconds over 10 trials, compared to the LLM model which takes an average of about 13.1 seconds over 10 trials (an average of about 13.09 seconds over 10 trials for LLM time + an average of about 0.04 seconds over 10 trials for Gurobi optimization time).

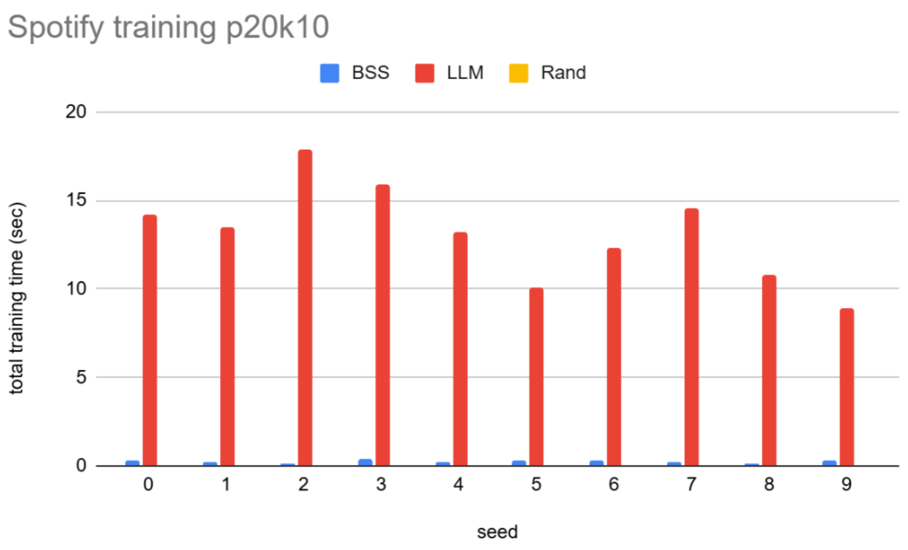


*Figure 4: Spotify total training time for 3 models on 10 seeds. For LLM, this is LLM call time + model train time. For BSS and Rand, this is just model train time.*

As we can see, the Gurobi optimization time lowered significantly between the models, due to lower search space; the bulk is of the time taken in this case is by the LLM calls. In a dataset with more features, this up-front time cost becomes acceptable and even preferable, as the optimization time becomes large due to a large search space. Note that Gurobi optimization is capped at a 60 second time limit in all cases unless otherwise specified.

To further demonstrate the feature selection capabilities, I calculated which features the LLM models chose the most for the Spotify dataset factoring the ranking given, and compared it to the Rand model and the linear correlation to target feature (table 10). Here we can see that although the LLM selection and correlation do not line up exactly, the LLM does a decent job of choosing relevant features while ignoring irrelevant features, especially compared to random selection.

Next, I tested this framework on a much higher dimensional dataset, this time predicting param2 (Hydrogen Bond Donor Number) from 353 other features. The models narrow the features down to 20 (when applicable), then select 10 or less to be used, just like before. From table 4, we can see that the LLM model again performs comparably to the BSS model, with the notable exception of the 6th trial, where it continues to perform well while BSS performs poorly. This trial is likely due to the BSS model running out of time to complete the optimization in that seed and returning a suboptimal solution, and also displays the robustness of the LLM model to these struggles. Meanwhile, Rand does extremely poorly, worse than both.

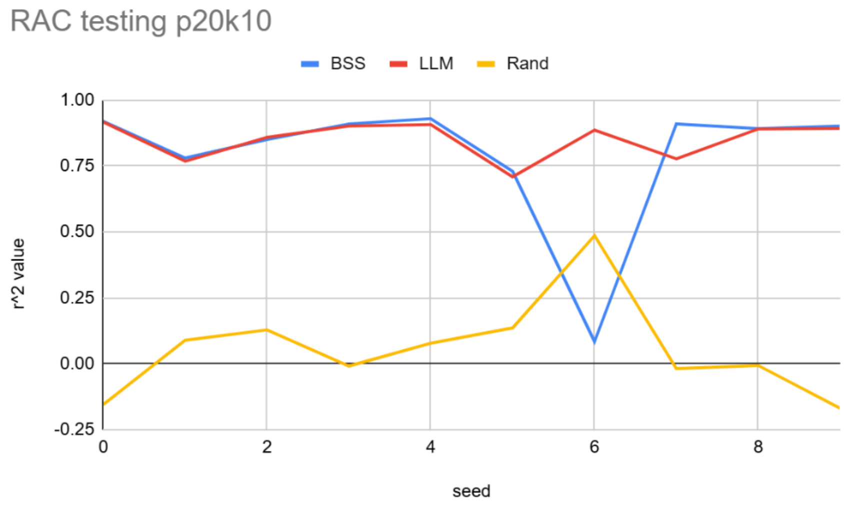


*Figure 5: RAC r^2 values for 3 models on 10 seeds.*

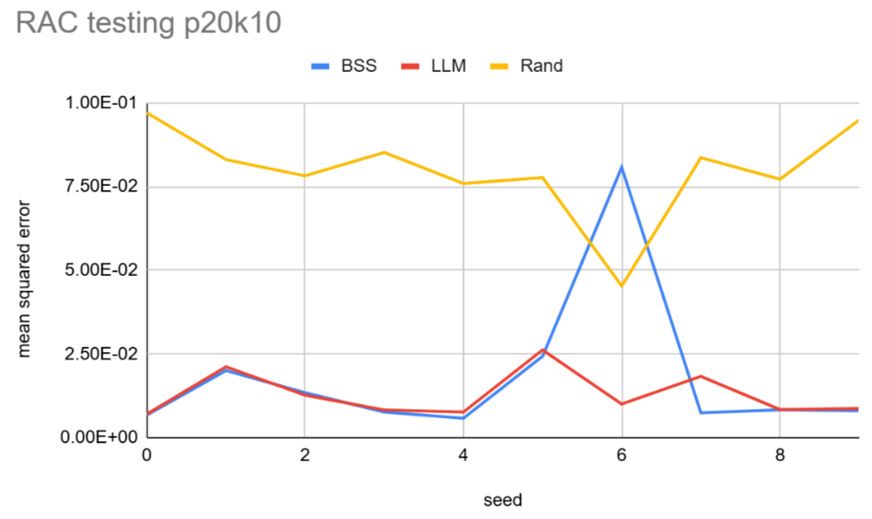


*Figure 6: RAC mean squared error for 3 models on 10 seeds.*

In table 3, we can see that the LLM model takes a bit longer, with an average of about 27 seconds to train, while the BSS now takes an average of about 60 seconds, presumably hitting the 60 second time limit on every trial, presumably due to its much larger search space.

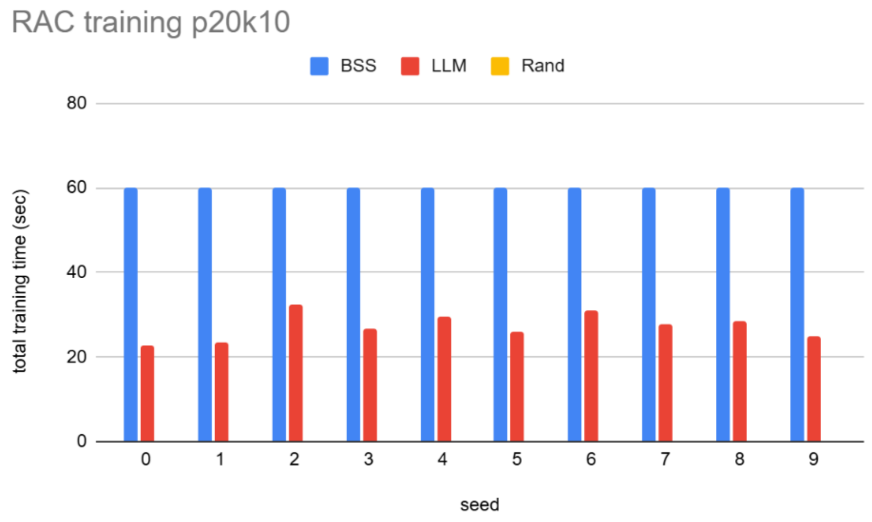


*Figure 7: RAC total training time for 3 models on 10 seeds.*

Now, with the RAC dataset, I found that the success of the LLM BSS is heavily dependent on a small number of features closely related to the target variable. It is important to mention that in this specific example, depending on the methods used to calculate the params, using these features in prediction may be data leakage; however, for our purposes, it shows that the LLM feature selection process is adept at selecting features highly associated with the target variable, which can then be extrapolated to future large-feature regression datasets.

## 3. Classification

After finding success with LLM BSS in regression, I moved on to classification by employing Support Vector Machines, or SVM. The SVM algorithm divides data into two assigned classes using a hyperplane in space with the same dimension as the number of features. An extension of this, the Soft Margin SVM, attempts to maximize the slack or margin between datapoints of opposite types, while allowing some datapoints to be misclassified if it benefits the overall model. The soft margins allow the SVM to classify non-linearly separable data and perform well even with noise in the data, or erroneous data that obscures the underlying pattern.

The Soft Margin SVM algorithm functions by minimizing the half the length of the weight vector squared plus the hyperparameter C times the sum of the slack variables (zeta), as in the following equation,

$$\underset{w,b}{\text{minimize}}\ \tfrac{1}{2}\|w\|^2 + C\textstyle\sum_{i=1}^{m}\zeta_i$$

along with the constraints;

$$y_i(w^T x_i + b) \geq 1 - \zeta_i \quad \text{and} \quad \zeta_i \geq 0 \quad \text{for } i = 1, 2, \ldots, m$$

A higher C value makes misclassification more expensive and results in a model with higher variance, or which is more tightly fit to the given data. A lower C value makes misclassification cheaper and leads to a model with higher bias, or which is less tightly fit to the given data.

In addition, I employed L0-norm regularization on this problem to limit the number of features. This is done by first creating a z vector of binary variables, each one corresponding to a feature. The weights are then constrained between their corresponding z value times and a positive and negative constant M, which encapsulates the weight values and causes the weight to be set to 0 if z is 0, essentially turning off that feature. Finally, z is constrained to be less than or equal to a given k value to enforce having only k or less features. This is shown with the following formulas;

$$-Mz_j \leq w_j \leq Mz_j \quad \forall j$$

$$\sum_{j=1}^{p} z_j \leq k$$

$$z_j \in \{0, 1\} \quad \forall j$$

Since the SVM with L0 regularization is limiting the number of features that can be used and trying to find the best subset of features to use, it could also be considered a form of best subset selection, extrapolated to solve classification problems. This allows it to tie in nicely with the regression best subset selection from earlier, to constitute another use of LLMs in best subset selection.

This LLM SVM framework operates in the same two step process as before, first calling the LLM to narrow down the features and second, using Gurobi to optimize the problem. The only difference is that now, instead of the regression best subset selection formulas from earlier, the problem is formulated to solve the Soft Margin SVM objectives and constraints.

I first tested this framework with a small dataset with 23 features besides the target variable, predicting whether a patient's glioma was benign or malignant. This dataset is relatively balanced, with around 42% of patients having lower grade glioma. For this small dataset, the LLM model performs slightly better than the Rand model with about 81% accuracy over 10 trials compared to 79% accuracy over 10 trials, but not quite as good as the SVM model on its own at about 87% accuracy over 10 trials. (table 6) Below are some graphs comparing the accuracy and roc score between the models:

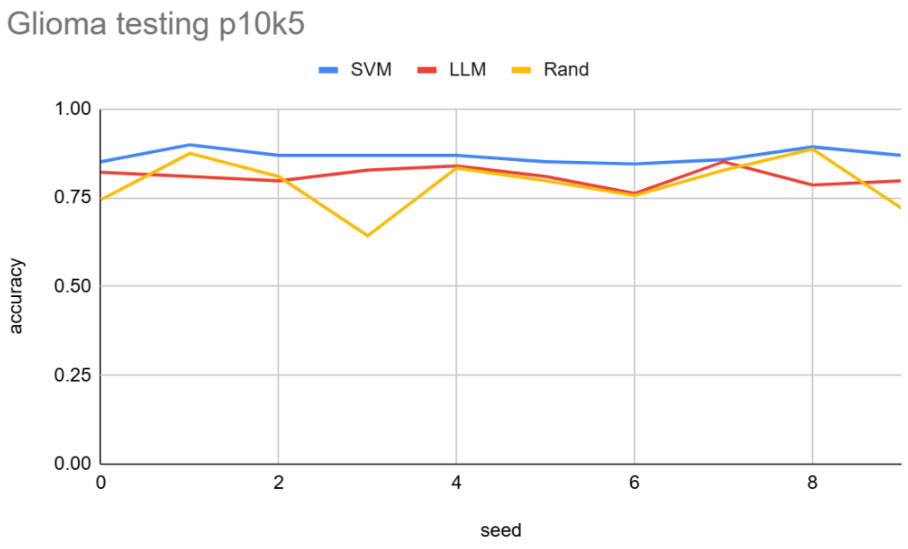


*Figure 8: Glioma accuracy for 3 models on 10 seeds.*

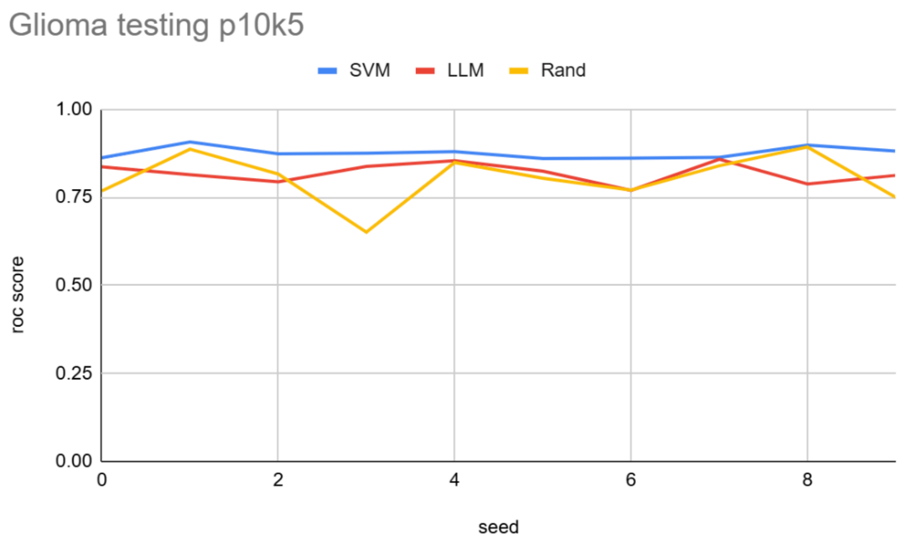


*Figure 9: Glioma roc score for 3 models on 10 seeds.*

Similar to the small regression dataset, the LLM model takes much longer to complete, when taking the time to call the LLM into account; the LLM call took an average of 23 seconds over 10 trials, while the SVM optimization for both the LLM model and the SVM alone took an average of less than a second over 10 trials. (table 5)

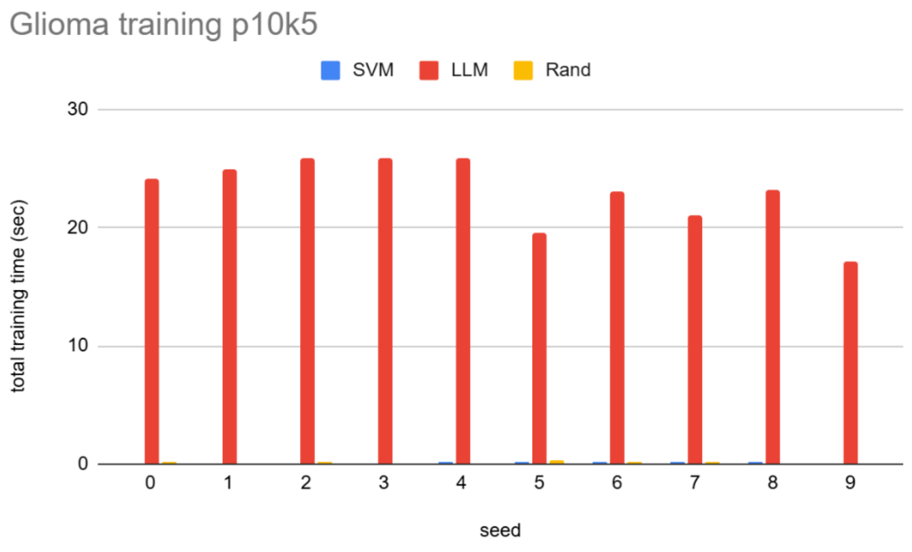


*Figure 10: Glioma total training time for 3 models on 10 seeds.*

Next, I moved on to the Parkinson’s dataset with 754 features, to predict from speech processing algorithm data whether or not a patient had Parkinson’s disease. This dataset was somewhat imbalanced, with about 75% of patients having the disease, so I included weights incorporated into the SVM framework to offset that imbalance and improve the f1 scores of the models, calculated with the following formula,

$$wj=n\_samples / (n\_classes * n\_samplesj)$$

where there were two classes. The variable n_samples is the number of samples of a specific class, and n_samplesj is the total number of samples. This formula assigns samples weights inversely proportional to their frequencies.

In this bigger dataset, both the LLM and the Rand models outperform the simple SVM model, as this one fails to come up with a strong model due to a high search space, only returning zeros for weights in every trial. The LLM model still does perform better than the Rand model, but only by slightly; both have about equal 80% accuracies over 10 trials, while the LLM model takes the lead in f1 score with an average of about .65 compared to .64, both over 10 trials.

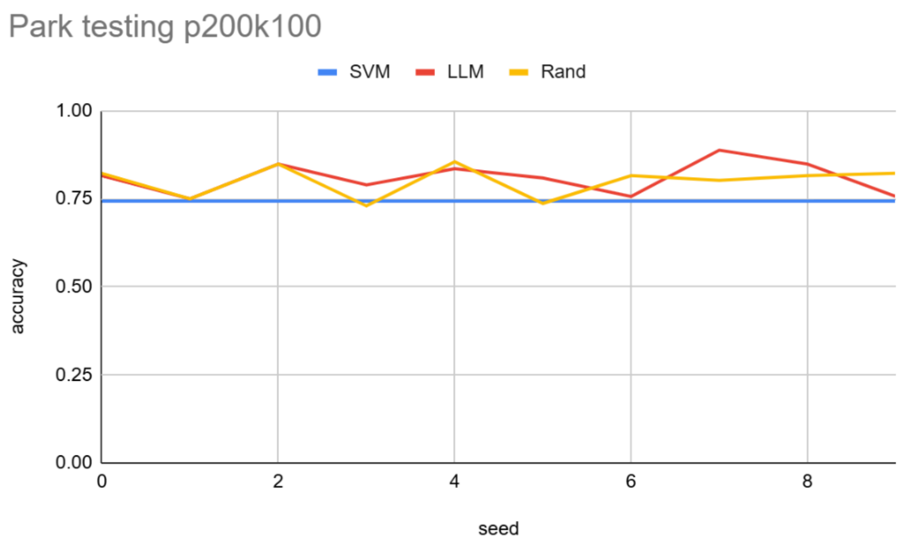


*Figure 11: Parkinson's accuracy for 3 models on 10 seeds.*

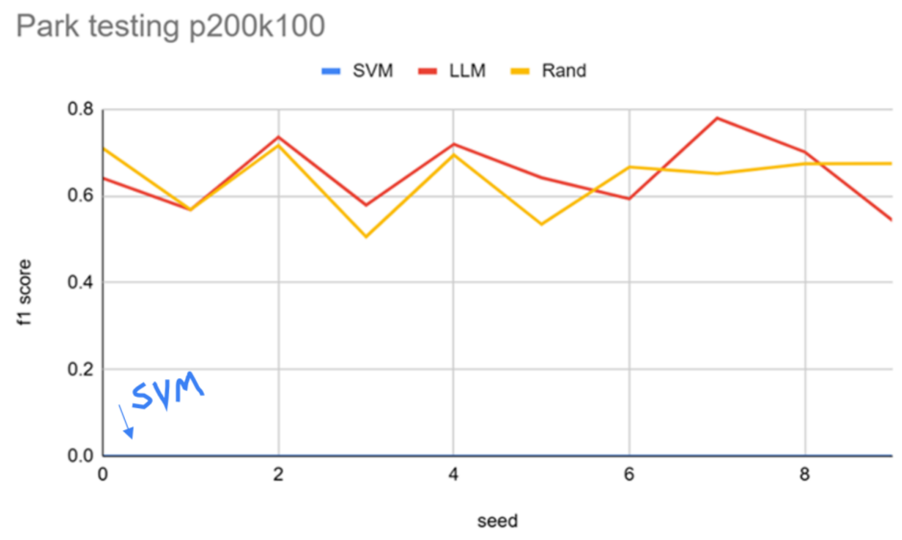


*Figure 12: Parkinson's f1 score for 3 models on 10 seeds.*

However, the LLM model takes a much longer time than the Rand one when adding up times; both seem to hit the 60 second time limit during Gurobi optimization, with the LLM additionally taking an average of 65 seconds over 10 trials to produce 100 ranked features.

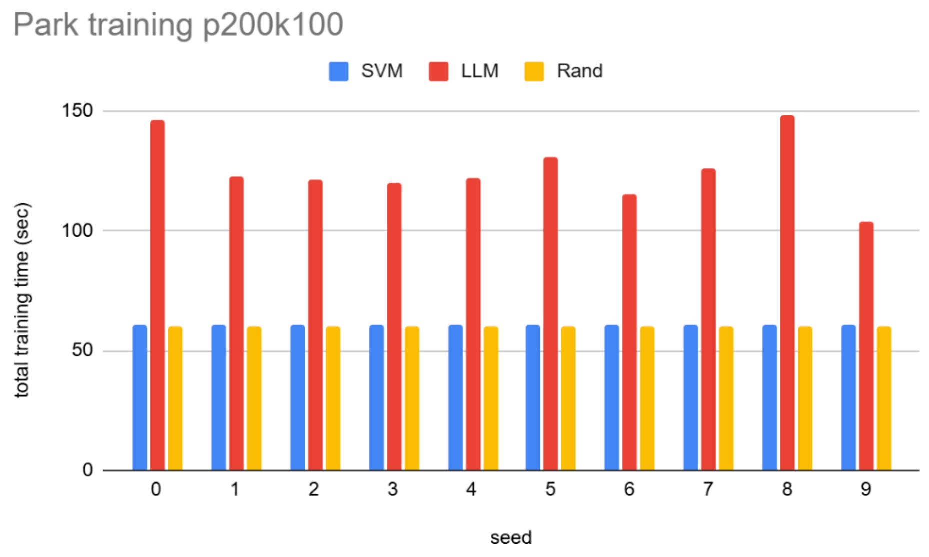


*Figure 13: Parkinson's total training time for 3 models on 10 seeds.*

Compared to LLMBSS, LLMSVM does not seem to improve significantly on the SVM model in these examples, only performing marginally better than random feature selection. In the future, I would suggest further research to cement the applicability of this model as better than random selection, possibly with grid searches and cross validation to find the optimal number of features to select and the optimal number of features to limit the optimizer to. In addition, improving upon context for specific problems or removing samples to make data balanced and reduce training time could improve performance significantly.

## 4.LLM Models

I began my research using api calls to Open AI, or ChatGPT, but switched to Gemini after some experimentation between models. As can be seen in table 11, the gemini-2.5-flash model is much more consistent than the gpt-3.5-turbo model. More testing needs to be done to ascertain what LLM is the best for this task; however, note that all the LLM data presented here is using the gemini-2.5-flash model to generate chosen feature names, due to its higher consistency.

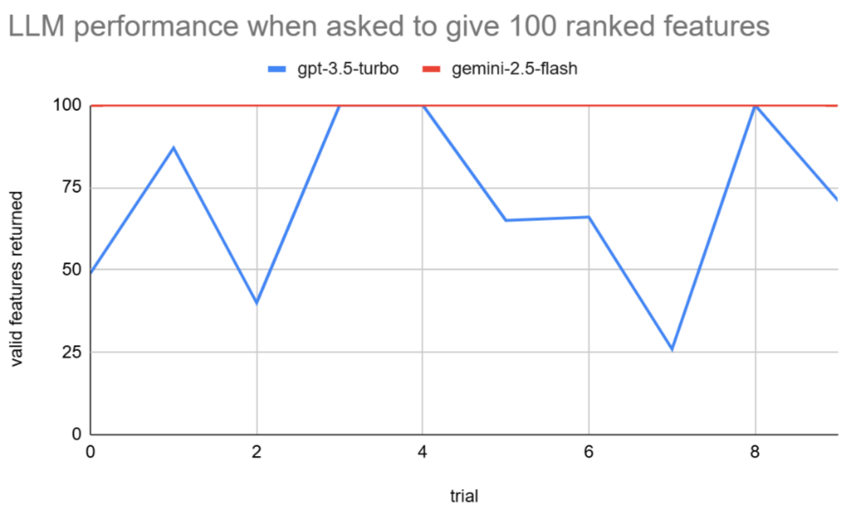


*Figure 14: valid features returned for 2 models on 10 trials.*

## 5. Conclusion

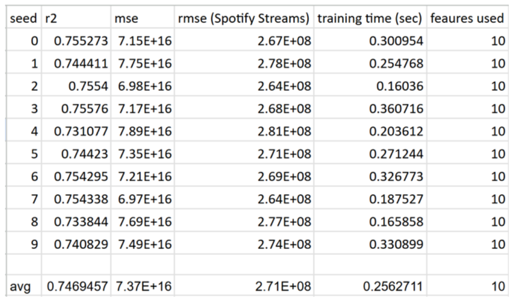
In conclusion, the LLMBSS framework is an innovative framework that combines recent advances ai in the form of large language models and mathematical best subset selection to produce interpretable and accurate machine learning models. As opposed to LLM Lasso, another framework that uses LLM feature selection, this framework does all its predictive work with best subset selection, which makes it much more accessible and explainable in situations where how a prediction is calculated is extremely important, such as business and insurance decisions. Which features are used and their weights are easily extractible, as the model simply uses an intercept and a coefficient vector with the same length of the number of features in the dataset to predict an outcome.

From the data, LLMBSS for regression is shown to perform well on small and big datasets, even outperforming BSS alone in the big dataset. The time taken for the LLM call is a significant downside in the small dataset, but becomes preferable in the large dataset. The LLMSVM framework only performed marginally better than random feature selection in both the small and big datasets, and is seen to have a much larger training time in the large dataset. More extensive testing is need for that framework to be viable.

In the future, I would suggest more extensive research with more datasets, each having cross validation and grid search to tune the hyperparameters: the number of features to ask the LLM to select (p), the number of features to select through best subset selection (k), and for the LLMSVM, the C and M values. These should be tuned to minimize time spent training and maximize performance metrics, like r^2 value and mean squared error for LLMBSS and accuracy, roc score, and f1 score for LLMSVM. In addition, different contexts for the LLM and data balance should be taken into account.

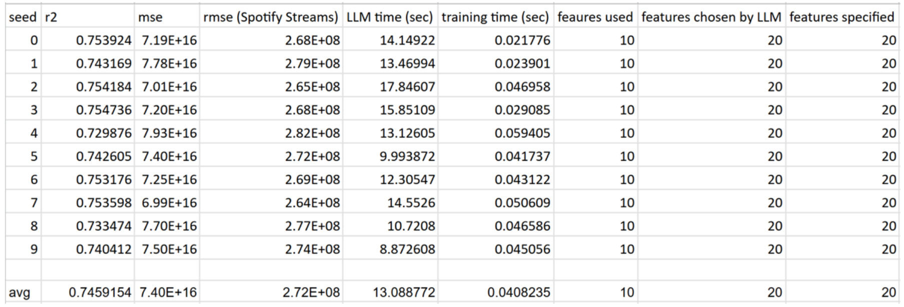
## 6. Appendix

**BSS Training**



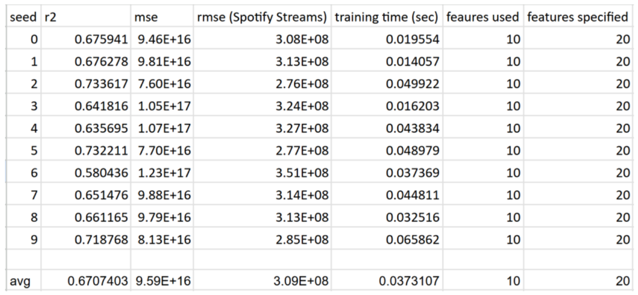
| seed | r2 | mse | rmse (Spotify Streams) | training time (sec) | feaures used |
|---|---|---|---|---|---|
| 0 | 0.755273 | 7.15E+16 | 2.67E+08 | 0.300954 | 10 |
| 1 | 0.744411 | 7.75E+16 | 2.78E+08 | 0.254768 | 10 |
| 2 | 0.7554 | 6.98E+16 | 2.64E+08 | 0.16036 | 10 |
| 3 | 0.75576 | 7.17E+16 | 2.68E+08 | 0.360716 | 10 |
| 4 | 0.731077 | 7.89E+16 | 2.81E+08 | 0.203612 | 10 |
| 5 | 0.74423 | 7.35E+16 | 2.71E+08 | 0.271244 | 10 |
| 6 | 0.754295 | 7.21E+16 | 2.69E+08 | 0.326773 | 10 |
| 7 | 0.754338 | 6.97E+16 | 2.64E+08 | 0.187527 | 10 |
| 8 | 0.733844 | 7.69E+16 | 2.77E+08 | 0.165858 | 10 |
| 9 | 0.740829 | 7.49E+16 | 2.74E+08 | 0.330899 | 10 |
| | | | | | |
| avg | 0.7469457 | 7.37E+16 | 2.71E+08 | 0.2562711 | 10 |

**LLM Training**



| seed | r2 | mse | rmse (Spotify Streams) | LLM time (sec) | training time (sec) | feaures used | features chosen by LLM | features specified |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.753924 | 7.19E+16 | 2.68E+08 | 14.14922 | 0.021776 | 10 | 20 | 20 |
| 1 | 0.743169 | 7.78E+16 | 2.79E+08 | 13.46994 | 0.023901 | 10 | 20 | 20 |
| 2 | 0.754184 | 7.01E+16 | 2.65E+08 | 17.84607 | 0.046958 | 10 | 20 | 20 |
| 3 | 0.754736 | 7.20E+16 | 2.68E+08 | 15.85109 | 0.029085 | 10 | 20 | 20 |
| 4 | 0.729876 | 7.93E+16 | 2.82E+08 | 13.12605 | 0.059405 | 10 | 20 | 20 |
| 5 | 0.742605 | 7.40E+16 | 2.72E+08 | 9.993872 | 0.041737 | 10 | 20 | 20 |
| 6 | 0.753176 | 7.25E+16 | 2.69E+08 | 12.30547 | 0.043122 | 10 | 20 | 20 |
| 7 | 0.753598 | 6.99E+16 | 2.64E+08 | 14.5526 | 0.050609 | 10 | 20 | 20 |
| 8 | 0.733474 | 7.70E+16 | 2.77E+08 | 10.7208 | 0.046586 | 10 | 20 | 20 |
| 9 | 0.740412 | 7.50E+16 | 2.74E+08 | 8.872608 | 0.045056 | 10 | 20 | 20 |
| | | | | | | | | |
| avg | 0.7459154 | 7.40E+16 | 2.72E+08 | 13.088772 | 0.0408235 | 10 | 20 | 20 |

**Rand Training**

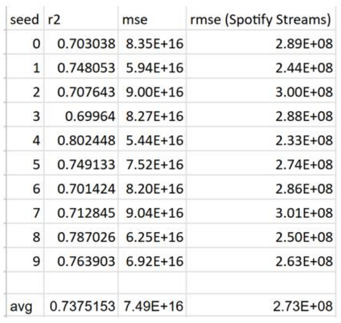


| seed | r2 | mse | rmse (Spotify Streams) | training time (sec) | feaures used | features specified |
|---|---|---|---|---|---|---|
| 0 | 0.675941 | 9.46E+16 | 3.08E+08 | 0.019554 | 10 | 20 |
| 1 | 0.676278 | 9.81E+16 | 3.13E+08 | 0.014057 | 10 | 20 |
| 2 | 0.733617 | 7.60E+16 | 2.76E+08 | 0.049922 | 10 | 20 |
| 3 | 0.641816 | 1.05E+17 | 3.24E+08 | 0.016203 | 10 | 20 |
| 4 | 0.635695 | 1.07E+17 | 3.27E+08 | 0.043834 | 10 | 20 |
| 5 | 0.732211 | 7.70E+16 | 2.77E+08 | 0.048979 | 10 | 20 |
| 6 | 0.580436 | 1.23E+17 | 3.51E+08 | 0.037369 | 10 | 20 |
| 7 | 0.651476 | 9.88E+16 | 3.14E+08 | 0.044811 | 10 | 20 |
| 8 | 0.661165 | 9.79E+16 | 3.13E+08 | 0.032516 | 10 | 20 |
| 9 | 0.718768 | 8.13E+16 | 2.85E+08 | 0.065862 | 10 | 20 |
| | | | | | | |
| avg | 0.6707403 | 9.59E+16 | 3.09E+08 | 0.0373107 | 10 | 20 |

Table 1: Spotify training results, where 20 features are specified and best subset selection choose 10 or less. The BSS model selects these 10 features from all the available features, while the others first narrow down the possible features through an LLM call and Random selection,

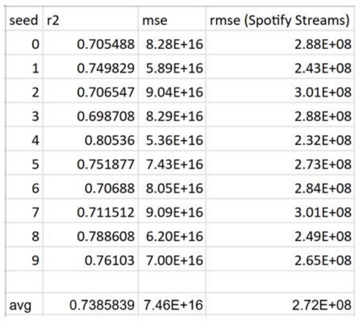
respectively. LLM time is the time taken for the API call to the LLM to return an answer, in this case Gemini.

**BSS Testing**



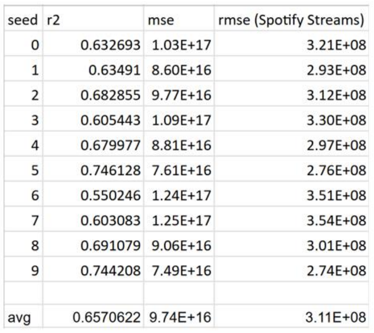
| seed | r2 | mse | rmse (Spotify Streams) |
|---|---|---|---|
| 0 | 0.703038 | 8.35E+16 | 2.89E+08 |
| 1 | 0.748053 | 5.94E+16 | 2.44E+08 |
| 2 | 0.707643 | 9.00E+16 | 3.00E+08 |
| 3 | 0.69964 | 8.27E+16 | 2.88E+08 |
| 4 | 0.802448 | 5.44E+16 | 2.33E+08 |
| 5 | 0.749133 | 7.52E+16 | 2.74E+08 |
| 6 | 0.701424 | 8.20E+16 | 2.86E+08 |
| 7 | 0.712845 | 9.04E+16 | 3.01E+08 |
| 8 | 0.787026 | 6.25E+16 | 2.50E+08 |
| 9 | 0.763903 | 6.92E+16 | 2.63E+08 |
| | | | |
| avg | 0.7375153 | 7.49E+16 | 2.73E+08 |

**LLM Testing**



| seed | r2 | mse | rmse (Spotify Streams) |
|---|---|---|---|
| 0 | 0.705488 | 8.28E+16 | 2.88E+08 |
| 1 | 0.749829 | 5.89E+16 | 2.43E+08 |
| 2 | 0.706547 | 9.04E+16 | 3.01E+08 |
| 3 | 0.698708 | 8.29E+16 | 2.88E+08 |
| 4 | 0.80536 | 5.36E+16 | 2.32E+08 |
| 5 | 0.751877 | 7.43E+16 | 2.73E+08 |
| 6 | 0.70688 | 8.05E+16 | 2.84E+08 |
| 7 | 0.711512 | 9.09E+16 | 3.01E+08 |
| 8 | 0.788608 | 6.20E+16 | 2.49E+08 |
| 9 | 0.76103 | 7.00E+16 | 2.65E+08 |
| | | | |
| avg | 0.7385839 | 7.46E+16 | 2.72E+08 |

**Rand Testing**

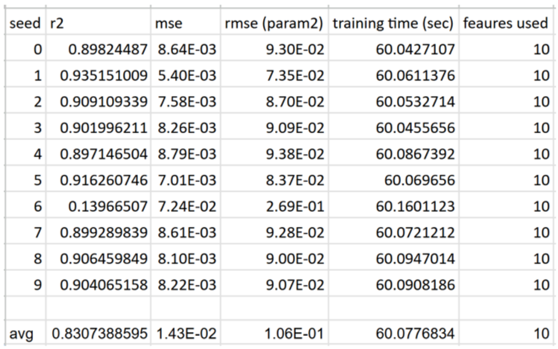


| seed | r2 | mse | rmse (Spotify Streams) |
|---|---|---|---|
| 0 | 0.632693 | 1.03E+17 | 3.21E+08 |
| 1 | 0.63491 | 8.60E+16 | 2.93E+08 |
| 2 | 0.682855 | 9.77E+16 | 3.12E+08 |
| 3 | 0.605443 | 1.09E+17 | 3.30E+08 |
| 4 | 0.679977 | 8.81E+16 | 2.97E+08 |
| 5 | 0.746128 | 7.61E+16 | 2.76E+08 |
| 6 | 0.550246 | 1.24E+17 | 3.51E+08 |
| 7 | 0.603083 | 1.25E+17 | 3.54E+08 |
| 8 | 0.691079 | 9.06E+16 | 3.01E+08 |
| 9 | 0.744208 | 7.49E+16 | 2.74E+08 |
| | | | |
| avg | 0.6570622 | 9.74E+16 | 3.11E+08 |

Table 2: Spotify testing results, from models trained in table 1.

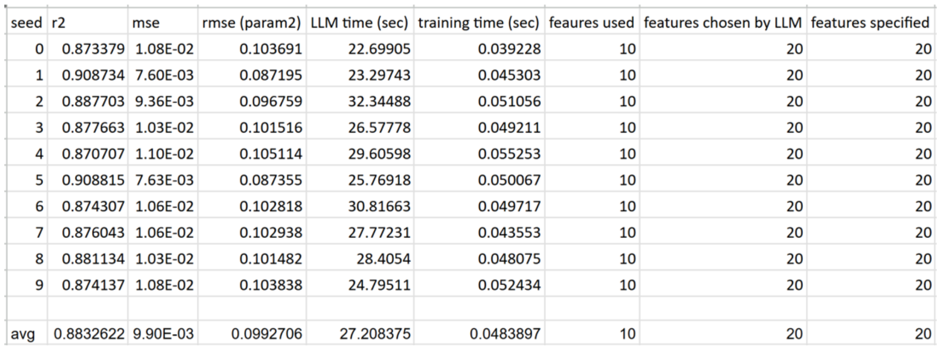
LLM Best Subset Selection

**BSS Training**



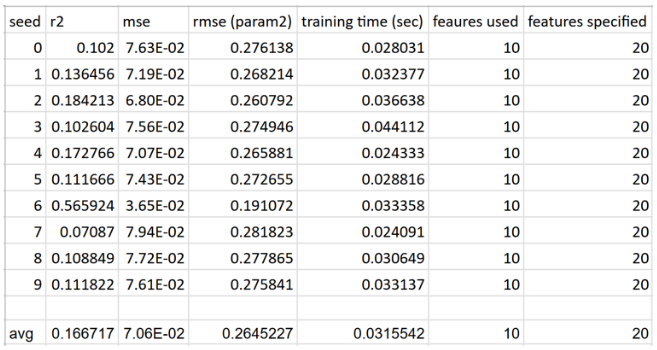
| seed | r2 | mse | rmse (param2) | training time (sec) | feaures used |
|---|---|---|---|---|---|
| 0 | 0.89824487 | 8.64E-03 | 9.30E-02 | 60.0427107 | 10 |
| 1 | 0.935151009 | 5.40E-03 | 7.35E-02 | 60.0611376 | 10 |
| 2 | 0.909109339 | 7.58E-03 | 8.70E-02 | 60.0532714 | 10 |
| 3 | 0.901996211 | 8.26E-03 | 9.09E-02 | 60.0455656 | 10 |
| 4 | 0.897146504 | 8.79E-03 | 9.38E-02 | 60.0867392 | 10 |
| 5 | 0.916260746 | 7.01E-03 | 8.37E-02 | 60.069656 | 10 |
| 6 | 0.13966507 | 7.24E-02 | 2.69E-01 | 60.1601123 | 10 |
| 7 | 0.899289839 | 8.61E-03 | 9.28E-02 | 60.0721212 | 10 |
| 8 | 0.906459849 | 8.10E-03 | 9.00E-02 | 60.0947014 | 10 |
| 9 | 0.904065158 | 8.22E-03 | 9.07E-02 | 60.0908186 | 10 |
| | | | | | |
| avg | 0.8307388595 | 1.43E-02 | 1.06E-01 | 60.0776834 | 10 |

**LLM Training**



| seed | r2 | mse | rmse (param2) | LLM time (sec) | training time (sec) | feaures used | features chosen by LLM | features specified |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.873379 | 1.08E-02 | 0.103691 | 22.69905 | 0.039228 | 10 | 20 | 20 |
| 1 | 0.908734 | 7.60E-03 | 0.087195 | 23.29743 | 0.045303 | 10 | 20 | 20 |
| 2 | 0.887703 | 9.36E-03 | 0.096759 | 32.34488 | 0.051056 | 10 | 20 | 20 |
| 3 | 0.877663 | 1.03E-02 | 0.101516 | 26.57778 | 0.049211 | 10 | 20 | 20 |
| 4 | 0.870707 | 1.10E-02 | 0.105114 | 29.60598 | 0.055253 | 10 | 20 | 20 |
| 5 | 0.908815 | 7.63E-03 | 0.087355 | 25.76918 | 0.050067 | 10 | 20 | 20 |
| 6 | 0.874307 | 1.06E-02 | 0.102818 | 30.81663 | 0.049717 | 10 | 20 | 20 |
| 7 | 0.876043 | 1.06E-02 | 0.102938 | 27.77231 | 0.043553 | 10 | 20 | 20 |
| 8 | 0.881134 | 1.03E-02 | 0.101482 | 28.4054 | 0.048075 | 10 | 20 | 20 |
| 9 | 0.874137 | 1.08E-02 | 0.103838 | 24.79511 | 0.052434 | 10 | 20 | 20 |
| | | | | | | | | |
| avg | 0.8832622 | 9.90E-03 | 0.0992706 | 27.208375 | 0.0483897 | 10 | 20 | 20 |

**Rand Training**

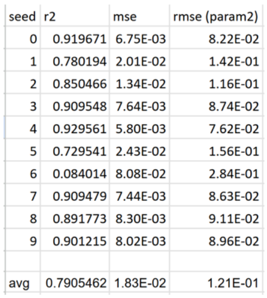


| seed | r2 | mse | rmse (param2) | training time (sec) | feaures used | features specified |
|---|---|---|---|---|---|---|
| 0 | 0.102 | 7.63E-02 | 0.276138 | 0.028031 | 10 | 20 |
| 1 | 0.136456 | 7.19E-02 | 0.268214 | 0.032377 | 10 | 20 |
| 2 | 0.184213 | 6.80E-02 | 0.260792 | 0.036638 | 10 | 20 |
| 3 | 0.102604 | 7.56E-02 | 0.274946 | 0.044112 | 10 | 20 |
| 4 | 0.172766 | 7.07E-02 | 0.265881 | 0.024333 | 10 | 20 |
| 5 | 0.111666 | 7.43E-02 | 0.272655 | 0.028816 | 10 | 20 |
| 6 | 0.565924 | 3.65E-02 | 0.191072 | 0.033358 | 10 | 20 |
| 7 | 0.07087 | 7.94E-02 | 0.281823 | 0.024091 | 10 | 20 |
| 8 | 0.108849 | 7.72E-02 | 0.277865 | 0.030649 | 10 | 20 |
| 9 | 0.111822 | 7.61E-02 | 0.275841 | 0.033137 | 10 | 20 |
| | | | | | | |
| avg | 0.166717 | 7.06E-02 | 0.2645227 | 0.0315542 | 10 | 20 |

Table 3: RAC training results. 20 features were specified for LLM and Rand, and best subset selection chose 10 or less. LLM used is Gemini.

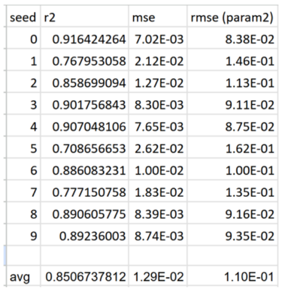
LLM Best Subset Selection

**BSS Testing**



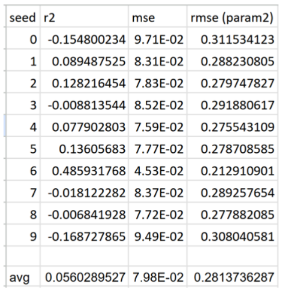
| seed | r2 | mse | rmse (param2) |
|---|---|---|---|
| 0 | 0.919671 | 6.75E-03 | 8.22E-02 |
| 1 | 0.780194 | 2.01E-02 | 1.42E-01 |
| 2 | 0.850466 | 1.34E-02 | 1.16E-01 |
| 3 | 0.909548 | 7.64E-03 | 8.74E-02 |
| 4 | 0.929561 | 5.80E-03 | 7.62E-02 |
| 5 | 0.729541 | 2.43E-02 | 1.56E-01 |
| 6 | 0.084014 | 8.08E-02 | 2.84E-01 |
| 7 | 0.909479 | 7.44E-03 | 8.63E-02 |
| 8 | 0.891773 | 8.30E-03 | 9.11E-02 |
| 9 | 0.901215 | 8.02E-03 | 8.96E-02 |
| | | | |
| avg | 0.7905462 | 1.83E-02 | 1.21E-01 |

**LLM Testing**



| seed | r2 | mse | rmse (param2) |
|---|---|---|---|
| 0 | 0.916424264 | 7.02E-03 | 8.38E-02 |
| 1 | 0.767953058 | 2.12E-02 | 1.46E-01 |
| 2 | 0.858699094 | 1.27E-02 | 1.13E-01 |
| 3 | 0.901756843 | 8.30E-03 | 9.11E-02 |
| 4 | 0.907048106 | 7.65E-03 | 8.75E-02 |
| 5 | 0.708656653 | 2.62E-02 | 1.62E-01 |
| 6 | 0.886083231 | 1.00E-02 | 1.00E-01 |
| 7 | 0.777150758 | 1.83E-02 | 1.35E-01 |
| 8 | 0.890605775 | 8.39E-03 | 9.16E-02 |
| 9 | 0.89236003 | 8.74E-03 | 9.35E-02 |
| | | | |
| avg | 0.8506737812 | 1.29E-02 | 1.10E-01 |

**Rand Testing**

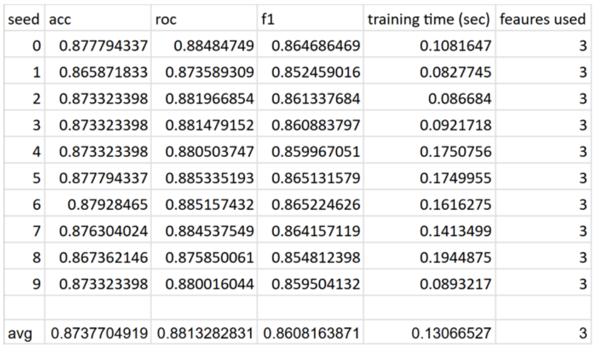


| seed | r2 | mse | rmse (param2) |
|---|---|---|---|
| 0 | -0.154800234 | 9.71E-02 | 0.311534123 |
| 1 | 0.089487525 | 8.31E-02 | 0.288230805 |
| 2 | 0.128216454 | 7.83E-02 | 0.279747827 |
| 3 | -0.008813544 | 8.52E-02 | 0.291880617 |
| 4 | 0.077902803 | 7.59E-02 | 0.275543109 |
| 5 | 0.13605683 | 7.77E-02 | 0.278708585 |
| 6 | 0.485931768 | 4.53E-02 | 0.212910901 |
| 7 | -0.018122282 | 8.37E-02 | 0.289257654 |
| 8 | -0.006841928 | 7.72E-02 | 0.277882085 |
| 9 | -0.168727865 | 9.49E-02 | 0.308040581 |
| | | | |
| avg | 0.0560289527 | 7.98E-02 | 0.2813736287 |

Table 4: RAC testing results, from models trained in table 3.

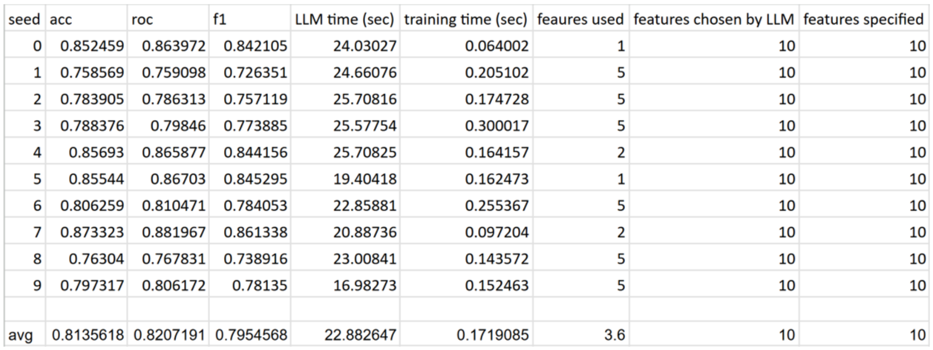
LLM Best Subset Selection

**SVM Training**



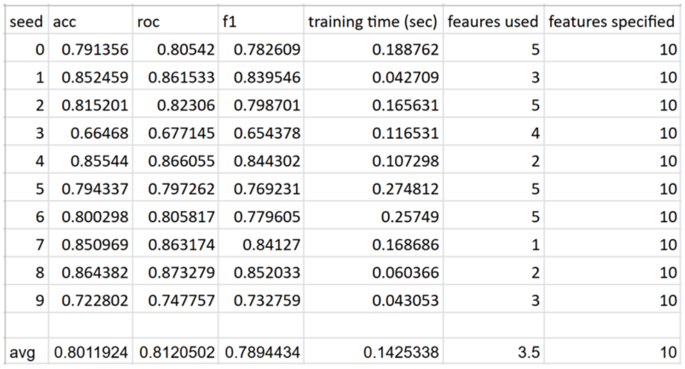
| seed | acc | roc | f1 | training time (sec) | feaures used |
|---|---|---|---|---|---|
| 0 | 0.877794337 | 0.88484749 | 0.864686469 | 0.1081647 | 3 |
| 1 | 0.865871833 | 0.873589309 | 0.852459016 | 0.0827745 | 3 |
| 2 | 0.873323398 | 0.881966854 | 0.861337684 | 0.086684 | 3 |
| 3 | 0.873323398 | 0.881479152 | 0.860883797 | 0.0921718 | 3 |
| 4 | 0.873323398 | 0.880503747 | 0.859967051 | 0.1750756 | 3 |
| 5 | 0.877794337 | 0.885335193 | 0.865131579 | 0.1749955 | 3 |
| 6 | 0.87928465 | 0.885157432 | 0.865224626 | 0.1616275 | 3 |
| 7 | 0.876304024 | 0.884537549 | 0.864157119 | 0.1413499 | 3 |
| 8 | 0.867362146 | 0.875850061 | 0.854812398 | 0.1944875 | 3 |
| 9 | 0.873323398 | 0.880016044 | 0.859504132 | 0.0893217 | 3 |
| | | | | | |
| avg | 0.8737704919 | 0.8813282831 | 0.8608163871 | 0.13066527 | 3 |

**LLM Training**



| seed | acc | roc | f1 | LLM time (sec) | training time (sec) | feaures used | features chosen by LLM | features specified |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.852459 | 0.863972 | 0.842105 | 24.03027 | 0.064002 | 1 | 10 | 10 |
| 1 | 0.758569 | 0.759098 | 0.726351 | 24.66076 | 0.205102 | 5 | 10 | 10 |
| 2 | 0.783905 | 0.786313 | 0.757119 | 25.70816 | 0.174728 | 5 | 10 | 10 |
| 3 | 0.788376 | 0.79846 | 0.773885 | 25.57754 | 0.300017 | 5 | 10 | 10 |
| 4 | 0.85693 | 0.865877 | 0.844156 | 25.70825 | 0.164157 | 2 | 10 | 10 |
| 5 | 0.85544 | 0.86703 | 0.845295 | 19.40418 | 0.162473 | 1 | 10 | 10 |
| 6 | 0.806259 | 0.810471 | 0.784053 | 22.85881 | 0.255367 | 5 | 10 | 10 |
| 7 | 0.873323 | 0.881967 | 0.861338 | 20.88736 | 0.097204 | 2 | 10 | 10 |
| 8 | 0.76304 | 0.767831 | 0.738916 | 23.00841 | 0.143572 | 5 | 10 | 10 |
| 9 | 0.797317 | 0.806172 | 0.78135 | 16.98273 | 0.152463 | 5 | 10 | 10 |
| | | | | | | | | |
| avg | 0.8135618 | 0.8207191 | 0.7954568 | 22.882647 | 0.1719085 | 3.6 | 10 | 10 |

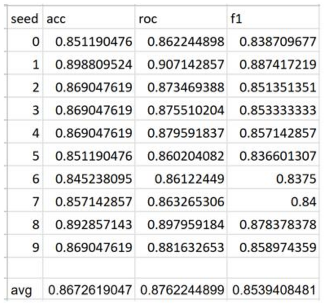
**Rand Training**



| seed | acc | roc | f1 | training time (sec) | feaures used | features specified |
|---|---|---|---|---|---|---|
| 0 | 0.791356 | 0.80542 | 0.782609 | 0.188762 | 5 | 10 |
| 1 | 0.852459 | 0.861533 | 0.839546 | 0.042709 | 3 | 10 |
| 2 | 0.815201 | 0.82306 | 0.798701 | 0.165631 | 5 | 10 |
| 3 | 0.66468 | 0.677145 | 0.654378 | 0.116531 | 4 | 10 |
| 4 | 0.85544 | 0.866055 | 0.844302 | 0.107298 | 2 | 10 |
| 5 | 0.794337 | 0.797262 | 0.769231 | 0.274812 | 5 | 10 |
| 6 | 0.800298 | 0.805817 | 0.779605 | 0.25749 | 5 | 10 |
| 7 | 0.850969 | 0.863174 | 0.84127 | 0.168686 | 1 | 10 |
| 8 | 0.864382 | 0.873279 | 0.852033 | 0.060366 | 2 | 10 |
| 9 | 0.722802 | 0.747757 | 0.732759 | 0.043053 | 3 | 10 |
| | | | | | | |
| avg | 0.8011924 | 0.8120502 | 0.7894434 | 0.1425338 | 3.5 | 10 |

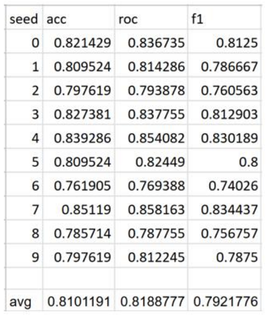
Table 5: Glioma training results. 10 features were specified for LLM and Rand, and best subset selection chose 5 or less. LLM used is Gemini.

**SVM Testing**



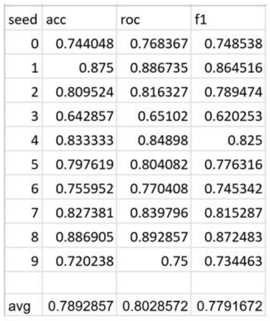
| seed | acc | roc | f1 |
|---|---|---|---|
| 0 | 0.851190476 | 0.862244898 | 0.838709677 |
| 1 | 0.898809524 | 0.907142857 | 0.887417219 |
| 2 | 0.869047619 | 0.873469388 | 0.851351351 |
| 3 | 0.869047619 | 0.875510204 | 0.853333333 |
| 4 | 0.869047619 | 0.879591837 | 0.857142857 |
| 5 | 0.851190476 | 0.860204082 | 0.836601307 |
| 6 | 0.845238095 | 0.86122449 | 0.8375 |
| 7 | 0.857142857 | 0.863265306 | 0.84 |
| 8 | 0.892857143 | 0.897959184 | 0.878378378 |
| 9 | 0.869047619 | 0.881632653 | 0.858974359 |
| | | | |
| avg | 0.8672619047 | 0.8762244899 | 0.8539408481 |

**LLM Testing**



| seed | acc | roc | f1 |
|---|---|---|---|
| 0 | 0.821429 | 0.836735 | 0.8125 |
| 1 | 0.809524 | 0.814286 | 0.786667 |
| 2 | 0.797619 | 0.793878 | 0.760563 |
| 3 | 0.827381 | 0.837755 | 0.812903 |
| 4 | 0.839286 | 0.854082 | 0.830189 |
| 5 | 0.809524 | 0.82449 | 0.8 |
| 6 | 0.761905 | 0.769388 | 0.74026 |
| 7 | 0.85119 | 0.858163 | 0.834437 |
| 8 | 0.785714 | 0.787755 | 0.756757 |
| 9 | 0.797619 | 0.812245 | 0.7875 |
| | | | |
| avg | 0.8101191 | 0.8188777 | 0.7921776 |

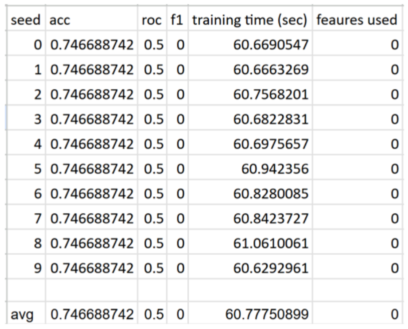
**Rand Testing**



| seed | acc | roc | f1 |
|---|---|---|---|
| 0 | 0.744048 | 0.768367 | 0.748538 |
| 1 | 0.875 | 0.886735 | 0.864516 |
| 2 | 0.809524 | 0.816327 | 0.789474 |
| 3 | 0.642857 | 0.65102 | 0.620253 |
| 4 | 0.833333 | 0.84898 | 0.825 |
| 5 | 0.797619 | 0.804082 | 0.776316 |
| 6 | 0.755952 | 0.770408 | 0.745342 |
| 7 | 0.827381 | 0.839796 | 0.815287 |
| 8 | 0.886905 | 0.892857 | 0.872483 |
| 9 | 0.720238 | 0.75 | 0.734463 |
| | | | |
| avg | 0.7892857 | 0.8028572 | 0.7791672 |

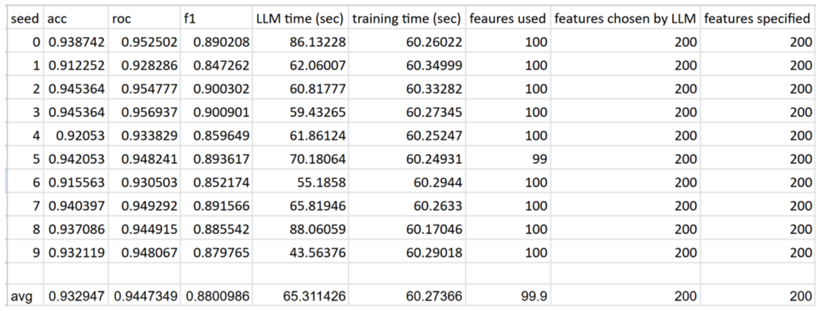
Table 6: Glioma testing results, from models trained in table 5.

**SVM Training**



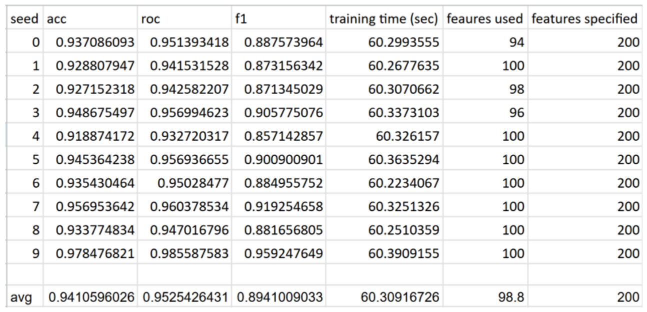
| seed | acc | roc | f1 | training time (sec) | feaures used |
|---|---|---|---|---|---|
| 0 | 0.746688742 | 0.5 | 0 | 60.6690547 | 0 |
| 1 | 0.746688742 | 0.5 | 0 | 60.6663269 | 0 |
| 2 | 0.746688742 | 0.5 | 0 | 60.7568201 | 0 |
| 3 | 0.746688742 | 0.5 | 0 | 60.6822831 | 0 |
| 4 | 0.746688742 | 0.5 | 0 | 60.6975657 | 0 |
| 5 | 0.746688742 | 0.5 | 0 | 60.942356 | 0 |
| 6 | 0.746688742 | 0.5 | 0 | 60.8280085 | 0 |
| 7 | 0.746688742 | 0.5 | 0 | 60.8423727 | 0 |
| 8 | 0.746688742 | 0.5 | 0 | 61.0610061 | 0 |
| 9 | 0.746688742 | 0.5 | 0 | 60.6292961 | 0 |
| | | | | | |
| avg | 0.746688742 | 0.5 | 0 | 60.77750899 | 0 |

**LLM Training**



| seed | acc | roc | f1 | LLM time (sec) | training time (sec) | feaures used | features chosen by LLM | features specified |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.938742 | 0.952502 | 0.890208 | 86.13228 | 60.26022 | 100 | 200 | 200 |
| 1 | 0.912252 | 0.928286 | 0.847262 | 62.06007 | 60.34999 | 100 | 200 | 200 |
| 2 | 0.945364 | 0.954777 | 0.900302 | 60.81777 | 60.33282 | 100 | 200 | 200 |
| 3 | 0.945364 | 0.956937 | 0.900901 | 59.43265 | 60.27345 | 100 | 200 | 200 |
| 4 | 0.92053 | 0.933829 | 0.859649 | 61.86124 | 60.25247 | 100 | 200 | 200 |
| 5 | 0.942053 | 0.948241 | 0.893617 | 70.18064 | 60.24931 | 99 | 200 | 200 |
| 6 | 0.915563 | 0.930503 | 0.852174 | 55.1858 | 60.2944 | 100 | 200 | 200 |
| 7 | 0.940397 | 0.949292 | 0.891566 | 65.81946 | 60.2633 | 100 | 200 | 200 |
| 8 | 0.937086 | 0.944915 | 0.885542 | 88.06059 | 60.17046 | 100 | 200 | 200 |
| 9 | 0.932119 | 0.948067 | 0.879765 | 43.56376 | 60.29018 | 100 | 200 | 200 |
| | | | | | | | | |
| avg | 0.932947 | 0.9447349 | 0.8800986 | 65.311426 | 60.27366 | 99.9 | 200 | 200 |

**Rand Training**

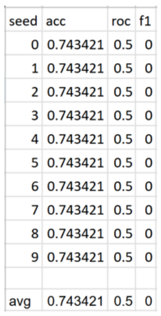


| seed | acc | roc | f1 | training time (sec) | feaures used | features specified |
|---|---|---|---|---|---|---|
| 0 | 0.937086093 | 0.951393418 | 0.887573964 | 60.2993555 | 94 | 200 |
| 1 | 0.928807947 | 0.941531528 | 0.873156342 | 60.2677635 | 100 | 200 |
| 2 | 0.927152318 | 0.942582207 | 0.871345029 | 60.3070662 | 98 | 200 |
| 3 | 0.948675497 | 0.956994623 | 0.905775076 | 60.3373103 | 96 | 200 |
| 4 | 0.918874172 | 0.932720317 | 0.857142857 | 60.326157 | 100 | 200 |
| 5 | 0.945364238 | 0.956936655 | 0.900900901 | 60.3635294 | 100 | 200 |
| 6 | 0.935430464 | 0.95028477 | 0.884955752 | 60.2234067 | 100 | 200 |
| 7 | 0.956953642 | 0.960378534 | 0.919254658 | 60.3251326 | 100 | 200 |
| 8 | 0.933774834 | 0.947016796 | 0.881656805 | 60.2510359 | 100 | 200 |
| 9 | 0.978476821 | 0.985587583 | 0.959247649 | 60.3909155 | 100 | 200 |
| | | | | | | |
| avg | 0.9410596026 | 0.9525426431 | 0.8941009033 | 60.30916726 | 98.8 | 200 |

Table 7: Park training results. 200 features were specified for LLM and Rand, and best subset selection chose 100 or less. LLM used is Gemini.

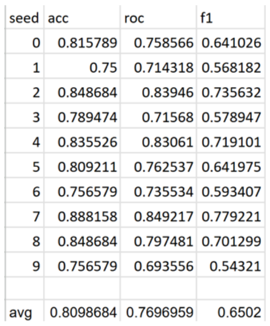
LLM Best Subset Selection

**SVM Testing**



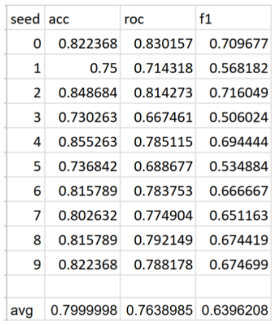
| seed | acc | roc | f1 |
|---|---|---|---|
| 0 | 0.743421 | 0.5 | 0 |
| 1 | 0.743421 | 0.5 | 0 |
| 2 | 0.743421 | 0.5 | 0 |
| 3 | 0.743421 | 0.5 | 0 |
| 4 | 0.743421 | 0.5 | 0 |
| 5 | 0.743421 | 0.5 | 0 |
| 6 | 0.743421 | 0.5 | 0 |
| 7 | 0.743421 | 0.5 | 0 |
| 8 | 0.743421 | 0.5 | 0 |
| 9 | 0.743421 | 0.5 | 0 |
| | | | |
| avg | 0.743421 | 0.5 | 0 |

**LLM Testing**



| seed | acc | roc | f1 |
|---|---|---|---|
| 0 | 0.815789 | 0.758566 | 0.641026 |
| 1 | 0.75 | 0.714318 | 0.568182 |
| 2 | 0.848684 | 0.83946 | 0.735632 |
| 3 | 0.789474 | 0.71568 | 0.578947 |
| 4 | 0.835526 | 0.83061 | 0.719101 |
| 5 | 0.809211 | 0.762537 | 0.641975 |
| 6 | 0.756579 | 0.735534 | 0.593407 |
| 7 | 0.888158 | 0.849217 | 0.779221 |
| 8 | 0.848684 | 0.797481 | 0.701299 |
| 9 | 0.756579 | 0.693556 | 0.54321 |
| | | | |
| avg | 0.8098684 | 0.7696959 | 0.6502 |

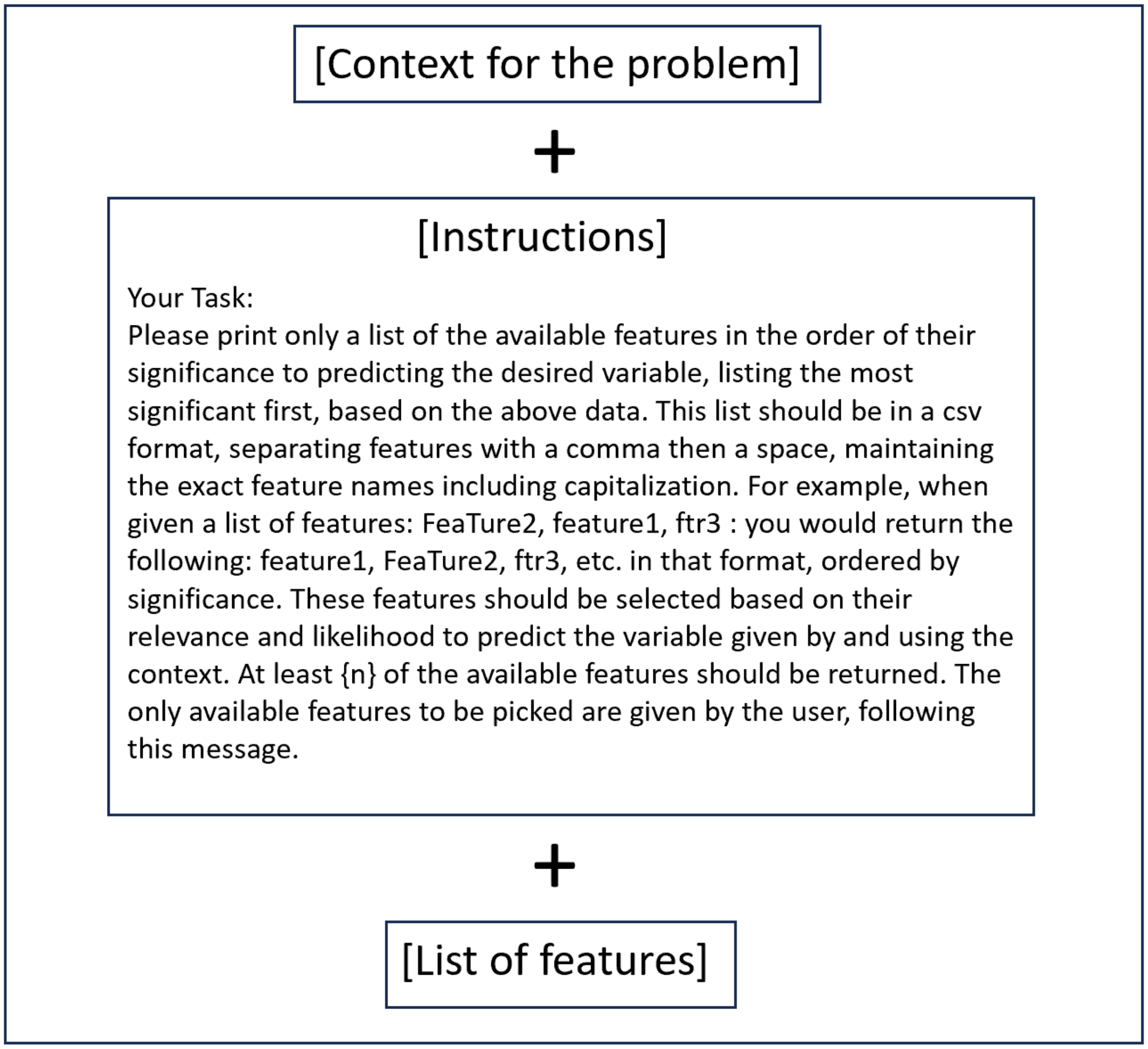
**Rand Testing**



| seed | acc | roc | f1 |
|---|---|---|---|
| 0 | 0.822368 | 0.830157 | 0.709677 |
| 1 | 0.75 | 0.714318 | 0.568182 |
| 2 | 0.848684 | 0.814273 | 0.716049 |
| 3 | 0.730263 | 0.667461 | 0.506024 |
| 4 | 0.855263 | 0.785115 | 0.694444 |
| 5 | 0.736842 | 0.688677 | 0.534884 |
| 6 | 0.815789 | 0.783753 | 0.666667 |
| 7 | 0.802632 | 0.774904 | 0.651163 |
| 8 | 0.815789 | 0.792149 | 0.674419 |
| 9 | 0.822368 | 0.788178 | 0.674699 |
| | | | |
| avg | 0.7999998 | 0.7638985 | 0.6396208 |

Table 8: Park testing results, from models trained in table 7.

Full message sent to LLM:



[Context for the problem]

+

[Instructions]

Your Task:
Please print only a list of the available features in the order of their significance to predicting the desired variable, listing the most significant first, based on the above data. This list should be in a csv format, separating features with a comma then a space, maintaining the exact feature names including capitalization. For example, when given a list of features: FeaTure2, feature1, ftr3 : you would return the following: feature1, FeaTure2, ftr3, etc. in that format, ordered by significance. These features should be selected based on their relevance and likelihood to predict the variable given by and using the context. At least {n} of the available features should be returned. The only available features to be picked are given by the user, following this message.

+

[List of features]

Table 9: The structure of the message sent to the LLM to request feature selection. “n” is the specified features to get. After retrieval, the response is split on commas and sliced to only include up to n features, as sometimes the LLM can return over or under the specified amount. Sometimes the LLM may return less features than specified, which is shown in the “features chosen by LLM” column of the training data when it occurs.

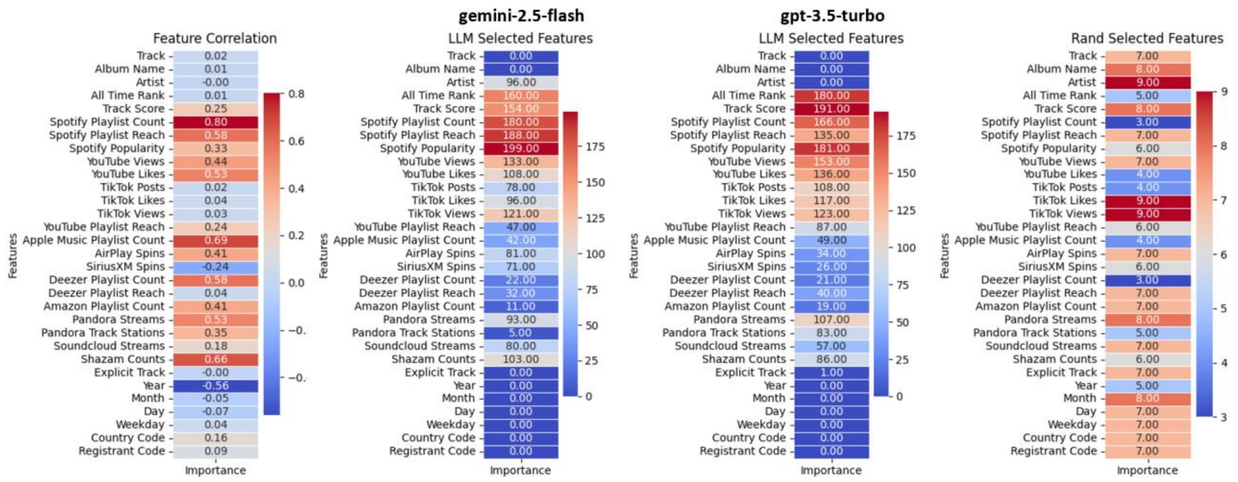


Table 10: Spotify feature correlation to target variable (Spotify Streams) heatmap compared to most selected features by LLMs, where 20 features were selected each trial, over 10 trials. Random feature selection is included for baseline, also with 20 features selected per trial, over 10 trials. LLM selected features was calculated over 10 trials by adding points for each feature, ranging up to 20 if they were ranked the highest, and down to 0 if the feature was not picked. Random Selected Features was calculated over 10 trials by adding 1 point if the feature was picked, and 0 if not.

| | (gpt-3.5-turbo) | (gemini-2.5-flash) | |
|---|---|---|---|
| trial | features chosen | features chosen | features specified |
| 0 | 49 | 100 | 100 |
| 1 | 87 | 100 | 100 |
| 2 | 40 | 100 | 100 |
| 3 | 100 | 100 | 100 |
| 4 | 100 | 100 | 100 |
| 5 | 65 | 100 | 100 |
| 6 | 66 | 100 | 100 |
| 7 | 26 | 100 | 100 |
| 8 | 100 | 100 | 100 |
| 9 | 71 | 100 | 100 |
| | | | |
| avg | 70.4 | 100 | 100 |

Table 11: Gemini and ChatGPT models compared when asked to give 100 ranked features from the Parkinson's dataset.

## 7. Works Cited

Daniel P. Jeong, Zachary C. Lipton, Pradeep Ravikumar, *LLM-Select: Feature Selection with Large Language Models*, 2024. URL https://arxiv.org/html/2407.02694v1

Erica Zhang, Ryunosuke Goto, Naomi Sagan, Jurik Mutter, Nick Phillips, Ash Alizadeh, Kangwook Lee, Jose Blanchet, Mert Pilanci, Robert Tibshirani, *LLM-Lasso: A Robust Framework for Domain-Informed Feature Selection and Regularization*, 2025. URL https://arxiv.org/pdf/2502.10648

Dimitris Bertsimas, Angela King, Rahul Mazumder, *Best subset selection via a modern optimization lens*, 2015. URL https://dspace.mit.edu/bitstream/handle/1721.1/108645/Bertsimas_Best%20subset%20selection.pdf

Gurobi Optimization, LLC, *Best Subset Selection: L0-Regression*, 2020. URL https://colab.research.google.com/github/Gurobi/modeling-examples/blob/master/linear_regression/l0_regression.ipynb#scrollTo=5_27Lz77SEy5

Claudio Escudero, *SVM example using Gurobi*, 2019. URL https://gist.github.com/escudero/912bd1ac94d6dee0b8624dcc81ef6650

Geeks for Geeks, *Support Vector Machine (SVM) Algorithm*, 2025. URL https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/

Anshul, *Support Vector Machine (SVM)*, 2025. URL https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

Kamaldeep, *How to Improve Class Imbalance using Class Weights in Machine Learning?*, 2025. URL https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/

Vincent Spruyt, *The Curse of Dimensionality in classification*, 2015. URL https://www.visiondummy.com/2014/04/curse-dimensionality-affect-classification/#google_vignette

## 8. Datasets

Most Streamed Spotify Songs 2024 https://www.kaggle.com/datasets/nelgiriyewithana/most-streamed-spotify-songs-2024/data

Metal-Organic Frame Materials Prediction (Referred to here as RAC): https://www.kaggle.com/datasets/marquis03/metal-organic-frame-materials-prediction?select=mof_round1_train_v2_230725 (I used RAC_train.csv)

Glioma Grading Clinical and Mutation Features https://archive.ics.uci.edu/dataset/759/glioma+grading+clinical+and+mutation+features+dataset

LLM Best Subset Selection

Parkinson's Disease Classification
https://archive.ics.uci.edu/dataset/470/parkinson+s+disease+classification