Abstract: The project aims to enhance feature selection in the data science pipeline by exploring the use of polynomial features in feature engineering for tabular data. Polynomial features involve raising original features to a power to create new features, which can capture nonlinear relationships between the features and the target variable. The effectiveness of this solution will be tested on at least 4 different datasets using a variety of machine learning models. The performance of models with and without polynomial features will be compared using metrics such as accuracy, precision, recall, and F1-score. Statistical analysis will be performed to determine if there is a significant difference in the performance of the models. The experiment will be repeated for different degrees of polynomial and types of datasets to assess how performance is affected.

Problem Description

The problem statement is to improve the feature selection element in the data science pipeline by exploring the use of polynomial features in feature engineering for tabular data.

- **Polynomial features:** Polynomial features are a type of feature engineering technique used to create new features by raising the original features to a power. This can be useful for handling non-linear relationships between the features and the target variable. There are also other advantages of using polynomial features like enhancing interpretability where the new features created by polynomial features can be more interpretable than the original features, which can help in understanding the relationship between the features and the target variable. They also increase the flexibility of the model by creating new features from the original features enabling it to fit complex patterns in the data.
    - Here's an example of how polynomial features can be used to handle non-linear relationships: Consider a dataset with two features, x1 and x2, and a target variable y.
    - A linear model, such as a linear regression, would model the relationship between x1 and x2 to y using a straight line. However, if there is a non-linear relationship between x1, x2, and y, a linear model may not be able to capture it. In this case, polynomial features can be used to create new features by raising x1 and x2 to a power.
    - For example, creating new features x1^2, x1*x2, x2^2 and so on. This can capture the non-linear relationship between x1, x2, and y. The new features can be used in a linear model, such as linear regression, to improve its performance.

Solution Overview

1. We will test the effectiveness/usefulness of the solution on 4 different datasets. A brief summary of the datasets used for running the models and comparison is as follows:
    - Dataset 1: The heart failure prediction daaset which was created by combining different datasets already available independently but not combined before. In this

dataset, over 11 common features predicting heart failureare avaiable for research purposes.

    - Source: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction

- Dataset 2: Bankruptcy data from the Taiwan Economic Journal for the years 1999–2009. The data were collected from the Taiwan Economic Journal for the years 1999 to 2009 and company bankruptcy was defined based on the business regulations of the Taiwan Stock Exchange.
    - Source: https://www.kaggle.com/datasets/fedesoriano/company-bankruptcy-prediction
- Dataset 3: The Vehicle Emissions and Smog Rating Classification dataset which has the amount of CO2 emissions by a vehicle depending on their various features. This dataset contains the official records of CO2 emissions data by various cars of different features. There are total of 26075 rows and 12 columns.
    - Source: https://www.kaggle.com/datasets/abhikdas2809/canadacaremissions
- Dataset 4: The forest cover type prediction dataset contains tree observations from four areas of the Roosevelt National Forest in Colorado. All observations are cartographic variables (no remote sensing) from 30 meter x 30 meter sections of forest. There are over half a million measurements total! This dataset includes information on tree type, shadow coverage, distance to nearby landmarks (roads etcetera), soil type, and local topography.
    - Source: https://www.kaggle.com/datasets/uciml/forest-cover-type-dataset

2. Then we will perform exploratory data analysis on the datasets to understand them better, followed by pre-processing and making each dataset ready for running the algorithms and downstream comparisons. The different steps followed in preparing and understanding the data is summarized sequentially here.

- **Exploratory Data Analysis:** Exploratory Data Analysis (EDA) is an essential step in any data science pipeline, including our current work on polynomial features. We will try to understand the relationship between each input variable we are feeding into the model and the target under consideration. This will help further in interpreting the different comparisons we draw between the raw features and transformed polynomial features as it will show us the kind of relationships (linear vs. non-linear) present in the data.
    - For this we will draw histograms for each independent feature, pair plots showing correlation pairwise of dependent variables with independent variables.
    - Understanding correlation structures in the data is important because some algorithms like logistic regression (we are doing classification) don't work well with multi-collinear datasets.

- **Data Processing Steps:** We do a series of steps to prepare the data for modeling.
    - **One hot encoding:** If categorical features are present in the data, we encode them with one hot encoding. The one hot encoded features are not considered for polynomial features.
    - **Scaling the numerical features:** We perform standard scaling on the numerical features.
    - **Resampling**: When the data set has more than 1000 records, we sample from both the classes (500 each) so that we can finish training the models within reasonable time due to computational limits.
    - **Multicollinearity detection:** Multicollinearity occurs when two or more independent variables in a regression model are highly correlated with each other, which can lead to instability in the model's coefficients and difficulty in interpreting the effects of individual variables on the target variable.
        - We detect multicollinearity using Variance Inflation Factor (VIF). The VIF measures how much the variance of the estimated regression coefficients is increased due to multicollinearity in the independent variables. VIF values greater than 1 indicate that multicollinearity may be present in the dataset, and values greater than 5 or 10 indicate a high degree of multicollinearity. We keep a cutoff of 5 for this work.
        - To calculate VIF, we first fit a linear regression model for each independent variable against all other independent variables. The VIF is then calculated as 1/(1-R^2), where R^2 is the coefficient of determination for the regression model. A VIF value of 1 indicates no correlation between the independent variables, while a value of greater than 1 indicates correlation.
    - **Feature Selection:** After removing the highly collinear features using VIF, if there are more than 15 features, we limit to top 15 correlated (with target) using Anova. We use f_classif as the feature selection method in scikit-learn that uses the ANOVA F-value to evaluate the significance of each feature in a classification problem. The ANOVA F-value measures the ratio of the variance between class means to the variance within each class.
        - In f_classif, the F-value is calculated for each feature, and a corresponding p-value is computed. The p-value indicates the probability of observing a F-value as extreme as the one computed by chance, assuming the null hypothesis that the feature has no effect on the target variable. Features with low p-values are considered to be more informative and are selected for inclusion in the final feature set.
    - **Creating polynomial features:** We will construct and repeat the experiment for different degrees of polynomial (2, 3) to check how the performance is affected with increasing degree of polynomial.

        - Going more than degree 3 is not advisable as it will explode the number of combinations and inturn features present in the input data we feed into the models.
    - **Train-Test Splitting:** We will be splitting each dataset into a training set and a test set (75 to 25 split). The training set will be used to train the machine learning models, while the test set will be used to evaluate their performance.

3. We then use a variety of machine learning models: We will use a variety of machine learning models to evaluate the effectiveness of polynomial feature expansion across different types of models. The following are the different steps under this.

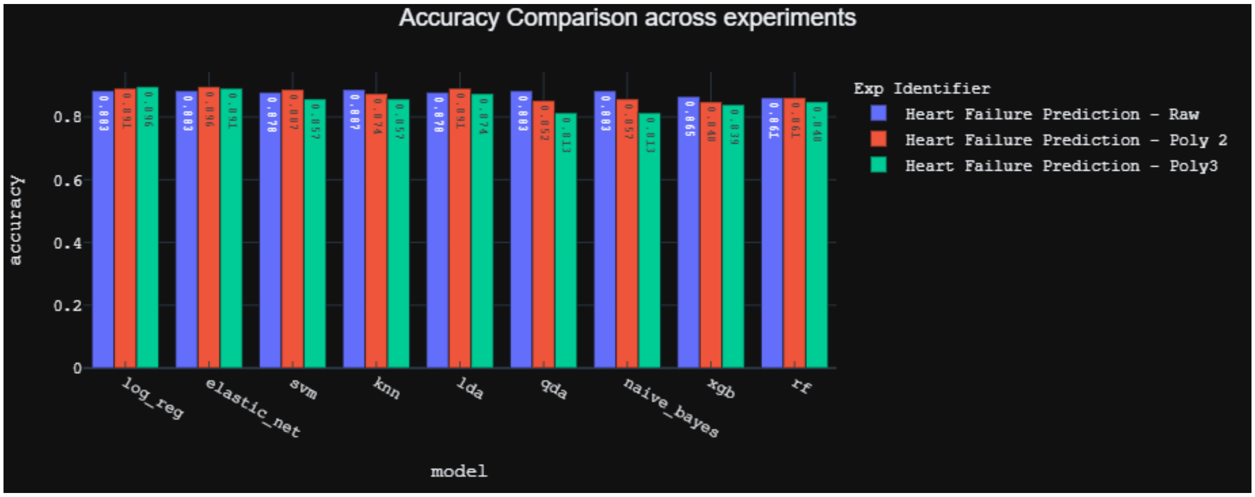
- **Model training:** We will use a variety of machine learning models to evaluate the effectiveness of polynomial feature expansion across different types of models.
    - **Models chosen:** Logistic regression, Elastic net classifier, Support Vector classifier, K-Nearest Neighbour classifier, Linear Discriminant Analysis classifier (LDA), Quadratic Discriminant Analysis (QDA) classifier, Naive Bayes classifier, Random forest classifier, XGBoost classifier
    - **Hyper-parameter tuning:** We tune the above selected models using bayesian hyper-optimization using cross validation of 5 folds. We use scikit-learn library for algorithms and hyperopt for tuning.
        - Hyperopt automates the process of hyperparameter optimization by defining a search space of possible hyperparameters and using Bayesian optimization algorithms to efficiently search this space.
        - The library provides several optimization algorithms, including Tree-structured Parzen Estimator (TPE) and Gaussian Process (GP) optimization, which are able to balance the exploration-exploitation tradeoff in a principled way.

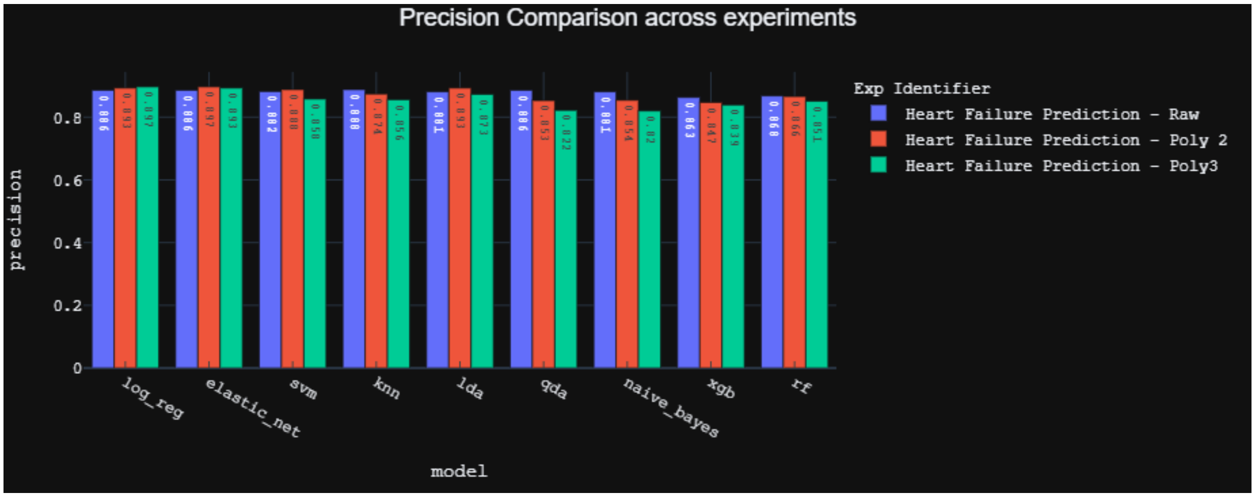
4. After training the models, we compare the performance of models with and without polynomial features.
    - **Evaluation:** We compare the performance of models with and without polynomial features. We will train the models both with and without polynomial features and compare their performance using metrics accuracy, precision, recall, and F1-score.
    - We will then perform analysis on the results to determine if there is a significant difference in the performance of the models with and without polynomial features.

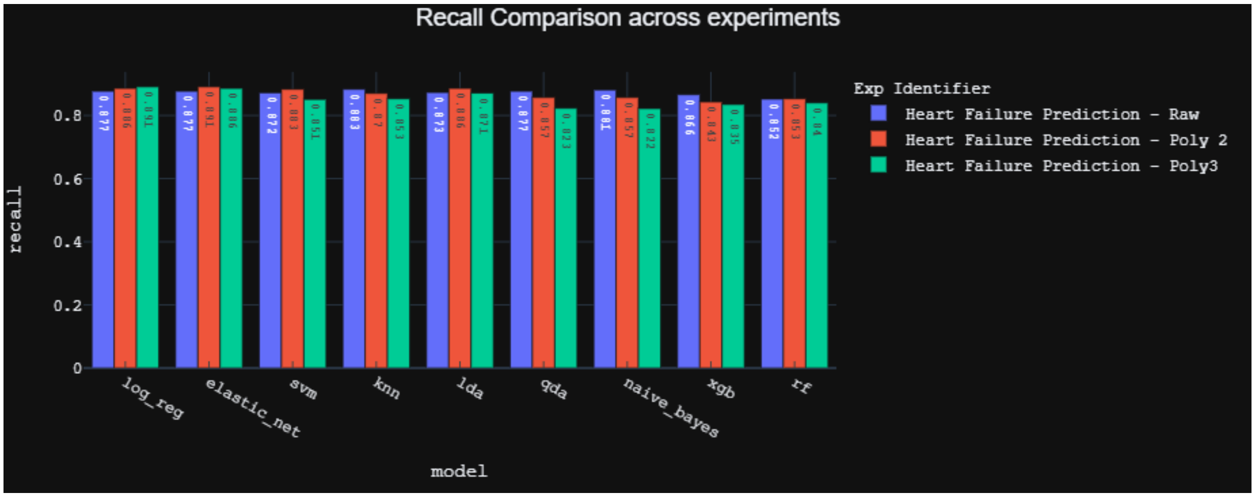
Experimental Evaluation

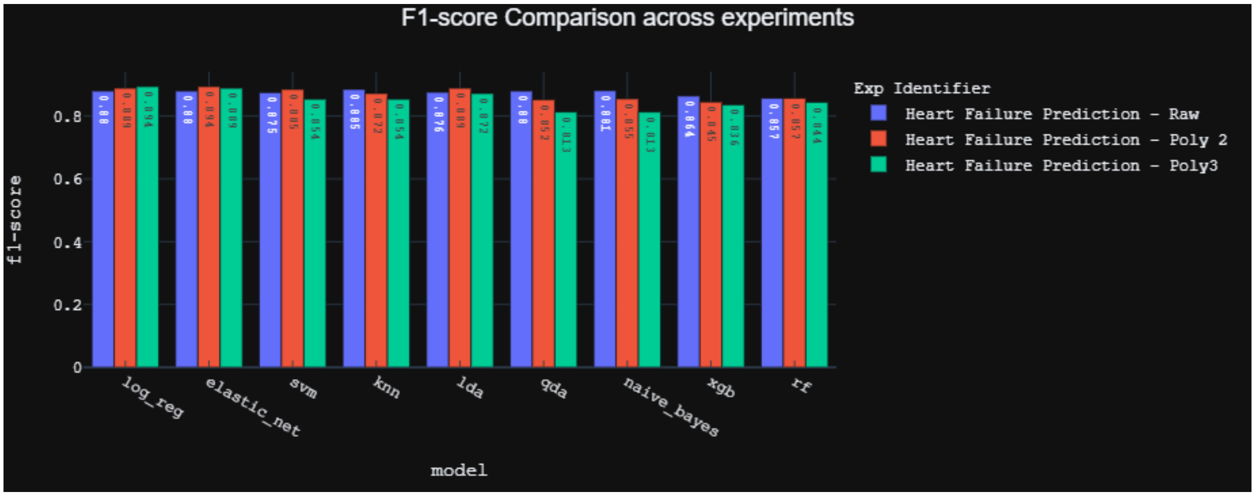
Here, we will provide the comparison graphs of metrics for each dataset for all the models we have run along with hyperparameter tuning.

- Dataset 1


Accuracy Comparison across experiments


Precision Comparison across experiments


Recall Comparison across experiments



<Download and Insert images for dataset 2>
<Download and Insert images for dataset 3>
<Download and Insert images for dataset 4>

Conclusion

- For dataset 1, we saw markable gains in the metrics for logistic regression and elastic net for both polynomials of degree 2 and 3, only better metrics for degree 2 in SVM (compared to raw features), no gains for rest of the models although the metrics are comparable. For dataset 2, polynomial features of degree 2 worked best for logistic regression and K-Nearest neighbors classifier, degree 3 for elastic net, XGBoost and Random Forest, in the rest of cases no reasonable difference is seen. In the third and fourth datasets, a trend similar to dataset 2 is observed.
- It is also evident that, even though accuracy numbers are comparable across raw and polynomial features, the improvement in precision, recall and F-score are high where there is a gain observed. Even though the gains are relatively small or moderate in some cases, even the small numbers translate to capturing false positives and false negatives and in turn translating to more perceived value in terms of using machine learning models across different applications.