UNIVERSITAT POLITÈCNICA DE CATALUNYA

FACULTAT D'INFORMÀTICA DE BARCELONA

Classification of Exoplanets

Machine Learning

Spring 2023

Authors:
Gega, Jezuela, *email*: jezuela.gega@estudiantat.upc.edu
Hoschek, Maren, *email*: maren.hoschek@estudiantat.upc.ed

Professors:
Arias Vicente, Marta
Coma Puig, Bernat

Contents

List of Figures

Chapter 1

Data exploration

1.1 Introduction

This document describes the implementation of various machine learning algorithms and their corresponding preprocessing techniques on the NASA Exoplanet dataset sourced from Kaggle. The dataset itself originates from the Kepler mission, a space-based observatory responsible for the discovery of numerous planets beyond our Solar System. For comprehensive field descriptions and additional details, the original data source can be found at CalTech. The primary objective of this project is to develop a robust machine learning model capable of accurately predicting whether a given observation represents a candidate for an exoplanet or not.

1.1.1 Data representation

To have a quick overview of our data first we do some data exploration to understand better. Firstly we check the data types of all our columns and to see better the distribution of each column we built an histogram for each column.

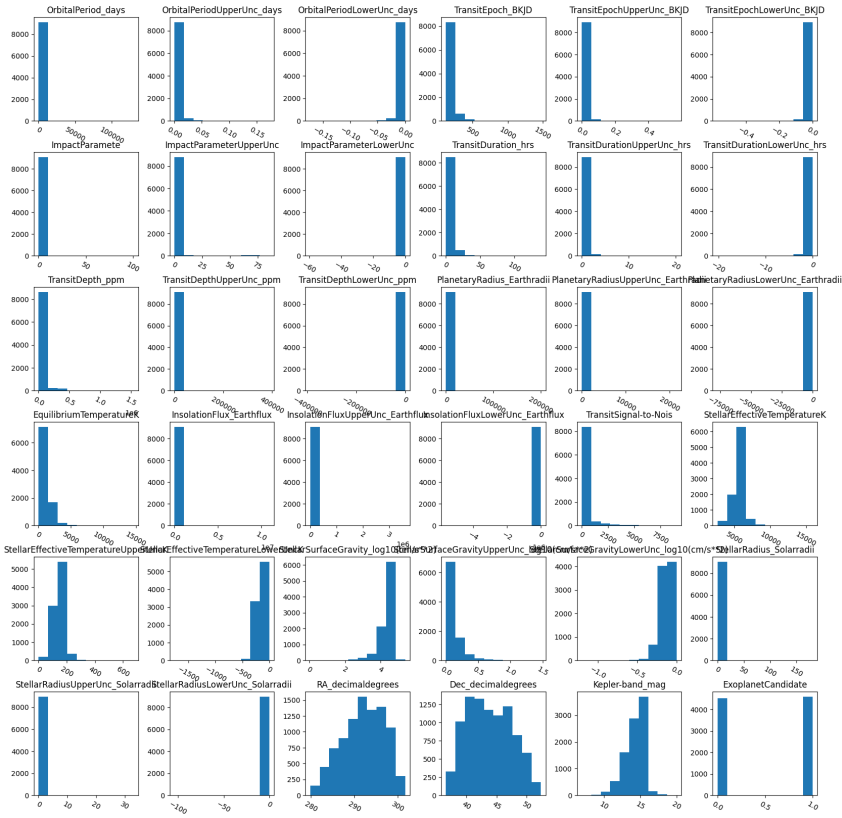


Figure 1.1: Histogram representation of each column

We saw from the graph that most of them do not have a specific (like normal for example) distribution, so in our analysis we will try to keep this detail in mind.

We checked our possible target values 'DispositionUsingKeplerData' and 'ExoplanetArchiveDisposition' but since 'ExoplanetArchiveDisposition' is data confirmed by humans and not collected we diecided to use DispositionUsingKeplerDataas our target, which makes our problem a binary one. We checked the balance of our data set and the target values and for 'DispositionUsingKeplerData' we have a it balanced:

- FALSE POSITIVE 4847
- CANDIDATE 4717

Since our data is balanced we are ready now to start polishing and cleaning it to make it ready to train.

Chapter 2

Preprocessing

2.1 Data Cleaning

Given our decision to use *DispositionUsingKeplerData* as the target variable for our analysis, our initial step involves creating a new column named *ExoplanetCandidate*. This column will encode the classes "candidate" and "false positive," which will be transformed into binary values. Specifically, we assign the value 1 to represent the "candidate" class and 0 to represent the "false positive" class. This binary representation is necessary to make the classes interpretable for machine learning models.

As the next step, we checked which columns we could drop from our dataset. Through an examination of their values and the descriptions provided by CalTech, we determined that these columns do not contribute any value to our analysis. An overview of the dropped columns can be found in appendix A.

2.2 Correlation

By performing correlation analysis on the Exoplanets dataset, we can uncover relationships and dependencies between different variables. Understanding the correlation between variables allows us to identify patterns, draw insights and understand variables that can make our dataset more or less biased towards a specific result.

As we can see from the matrix representation in figure 2.1 'DispositionScore' is highly correnlated with our target value 'ExoplanetCandidate' that is why we decide to drop this variable from our dataset. as well as we see other variable that are highly correlated or negatively correlated especially near the diagonal. Those relationships present risk in our analyses, that is why in further cleaning and pre-processing of the data we take measures like dropping or doing feature combinations in the next sections.

2.3 Missing Values

The majority of our dataset's columns contains missing values. Notably, the columns *EquilibriumTemperatureUpperUncK* and *EquilibriumTemperatureLowerUncK* exclusively contain null values, so they don't add any information to our analysis. Consequently, we have made the decision to drop these columns from our dataset.

Upon closer examination of the numerical data, certain patterns have emerged, indicating a correlation between missing values. Specifically for most of the uncertainty columns with missing values it seems, when the positive uncertainty is missing, the corresponding negative uncertainty is also absent. This observation has led us to employ a counting mechanism to determine the number of missing values per row.

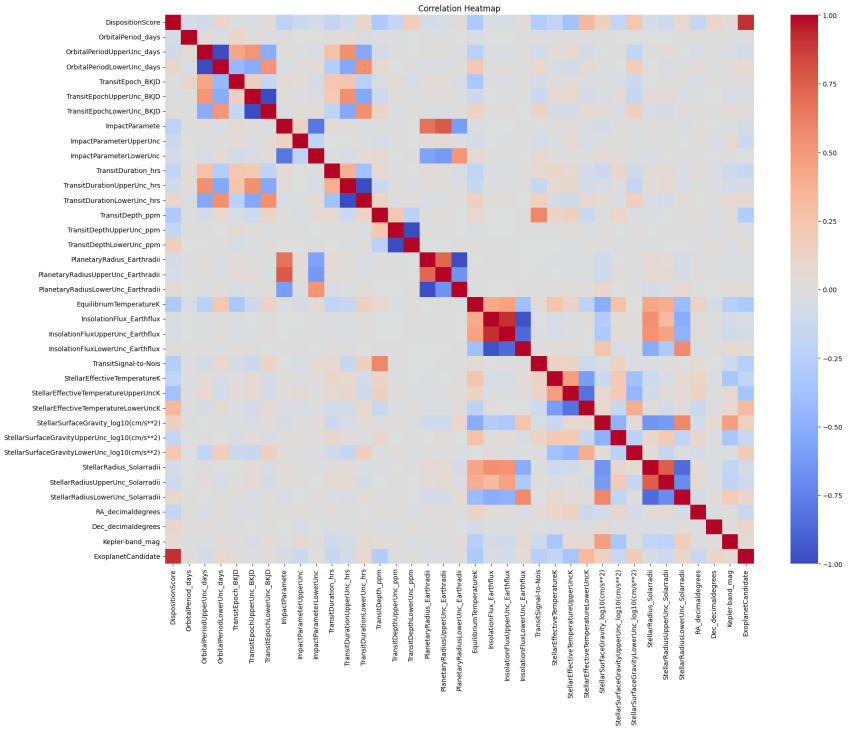


Figure 2.1: Correlation heatmap

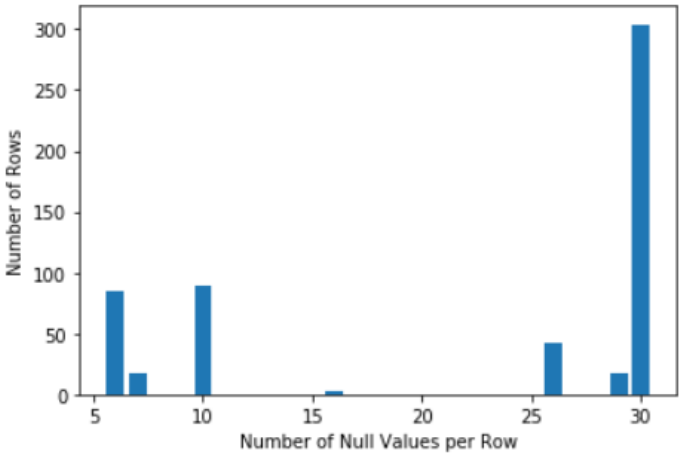


Figure 2.2: Number of missing values per row

Our findings reveal that some rows have as many as 30 null values. To ensure the reliability of our algorithm and to prevent an excessive proportion of imputed values (over 25%) that may introduce confusion rather than information, we have chosen to discard any rows containing more than 10 missing values. This process has resulted in a reduction from an initial count of 9564 rows to 9110 rows, equating to less than 5% data loss.

Subsequently, after this row elimination step, we recounted the presence of missing values and found that only seven columns still contain missing data. For these remaining columns, we applied the k-nearest neighbors (kNN) imputation technique with k=3. For the imputation we

used the kNN imputer from scikit-learn[1]. Importantly, the imputation process was applied after splitting the dataset into training and testing subsets, ensuring that nearest neighbors are sought exclusively within the respective train or test data space.[PV18] We choose kNN imputation over fixed value imputation methods like median or average due to the lack of normality assumptions in our data and our limited expertise in astronomy to determine suitable fixed values. KNN imputation's flexibility in handling different feature spaces makes it a better choice for our diverse dataset.[PV18]

2.4 Outliers

To check for outliers we first start with a representation of the columns in boxplots to see if there are outliers. Since the data doesn't follow a normal distribution, we will calculate the outlier data points using the statistical method called interquartile range (IQR) instead of using Z-score. Using the IQR, the outlier data points are the ones falling below Q1–1.5 IQR or above Q3 + 1.5 IQR. The Q1 is the 25th percentile and Q3 is the 75th percentile of the dataset, and IQR represents the interquartile range calculated by Q3 minus Q1 (Q3–Q1).

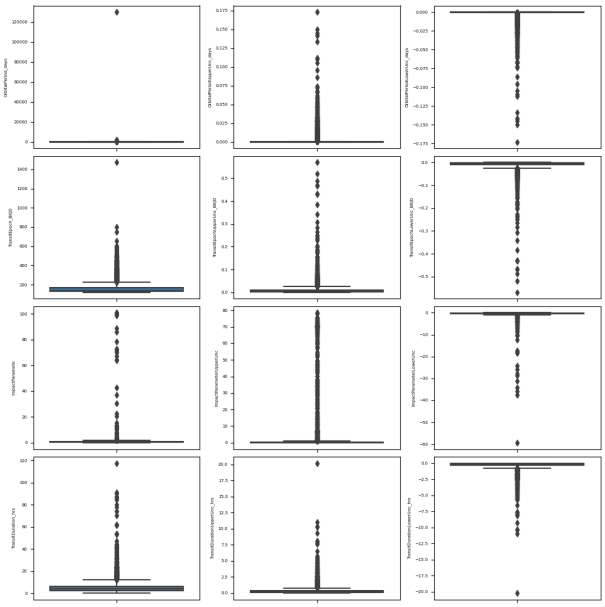


Figure 2.3: Representing outliers

We see that we have several outliers based on the method we used so we decided to smoth this outlier by using winsorization method. For this algorithm we set the thresholds 5

2.5 Feature Engineering

After completing the initial data cleaning process, our dataset still contains 34 features, which is a considerable number of features relative to the remaining 9110 rows. Through correlation analysis, we discovered a strong negative correlation among the uncertainty features. However, to ensure our analysis is not influenced by highly correlated features, we made the decision to consolidate the positive and negative uncertainty values into a single uncertainty feature. This consolidation was achieved by summing the absolute values of the two corresponding columns.

[1] https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html

By taking the absolute value, we aim to obtain an overall measure of uncertainty irrespective of its direction.

Example:
Looking at the *OrbitalPeriod_days* we have the positive uncertainty *OrbitalPeriodUpperUnc_days* and the negative uncertainty *OrbitalPeriodLowerUnc_days*. When now we have a *OrbitalPeriodUpperUnc_days* of 1.34 days and a *OrbitalPeriodLowerUnc_days* of -2.23 days our total uncertainty should be 3,57 and not -0,89 days therefore we take the absolute values.

After this feature engineering step was undertaken we reduced the dimensionality of our data to 25 features.

2.6 Scaling

In our dataset the features have a wide range of measurement units, including for example temperature, days, or gravity. This variance in measurement units introduces potential confusion for our model due to the differing value ranges.[Cho20] To address this issue, we applied data scaling techniques to our data. Since we cannot assume a Gaussian distribution for all features standardization methods are not be suitable our dataset. Instead, we opt for a Min/Max scaler, which transforms the values to a range between 0 and 1. This approach ensures that all features are on a consistent scale, mitigating the impact of different measurement units.[Nai22] For the scaling process, we utilize the Min/Max scaler provided by scikit-learn [2]. Notably, prior to scaling, we performed outlier handling to ensure the scaler's effectiveness is not compromised by outliers. It is important to note that the scaler is fitted solely on the training dataset, and the test dataset is scaled using the scaler derived from the training data. This approach prevents any transfer of knowledge from the training set to the test set.(ref)

[2] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

Chapter 3

Models

We selected logistic regression, decision trees, random forest, and support vector machines (SVM) as models to try on our dataset as they are all suitable for binary classification. Logistic regression was considered for its interpret-ability and simplicity. Decision trees were utilized to capture non-linear relationships and identify important features. Random forest was employed to mitigate over fitting and handle noisy data. Finally, support vector machines were used to leverage non-linear decision boundaries and handle datasets with high-dimensional features. By utilizing this combination of models, we aimed to explore different algorithmic approaches and leverage their respective advantages to obtain the most accurate and reliable predictions for the exoplanet classification.

3.1 Logistic Regression

Logistic regression is a statistical method used to model the relationship between a binary outcome variable and one or more independent variables. It estimates the probability of the outcome variable belonging to a specific category, given the values of the independent variables.[SW17]

After completing the prepossessing stage, our dataset still contains 25 features. To prevent our logistic regression model from becoming overly complex and over fitting to the training data, we want to introduce regularization to our model. We applied the logistic regression model provided by the scikit-learn library [1]. This model offers the hyperparameter 'C' to control the regularization strength. Since the optimal 'C' value for our specific data is unknown, we will utilize the Logistic Regression with Cross Validation (*LogisticRegressionCV*) model, also available in scikit-learn [2]. This logistic regression model incorporates built-in cross-validation to automatically determine the most suitable hyperparameters. In our case, we applied it to identify the optimal value for 'C'. Once we have determined the optimal 'C', we constructed a logistic regression model using this optimal value and performed cross-validation to evaluate the model's performance using various classification quality metrics. To be able to use the LogisticRegressionCV model, we needed to increase the maximum iteration parameter to 5000 (the default value is 100). The optimal C for our dataset is 21.544. Upon computing a logistic regression model with this C and evaluating its quality with cross validation, we obtained the results shown in 3.1.

The results obtained from the various cross-validation folds and evaluation metrics are very consistent, as they constantly hover around 83%. That the results across the different evaluation metrics are very similar means that the model is achieving a balanced performance in terms of classifying positive (Exoplanet candidate) and negative (false positive) instances.[Pin21] Our logistic regression model uses the following regression coefficients.

[1] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[2] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegressionCV.html

	fit_time	score_time	test_accuracy	test_f1_macro	test_precision_macro	test_recall_macro
0	0.190568	0.005455	0.837019	0.836624	0.839663	0.836765
1	0.144687	0.000000	0.830467	0.830430	0.830547	0.830403
2	0.152110	0.000000	0.829648	0.829622	0.829682	0.829602
3	0.152539	0.000000	0.828689	0.828314	0.830902	0.828418
4	0.170688	0.006996	0.832787	0.832528	0.834290	0.832565

Figure 3.1: Results of logistic regression

array([[-2.09145305, -0.1744613 , -0.96781113, -0.72096287, -9.39261285,-6.06647004, -5.87842029, -1.01877919, -0.80194291, 2.1679141 ,-3.45556392, -0.24651949, -0.70512551, 0.79616408, 1.41034958, 0.75683502, -0.20079409, -0.81050781, -2.5470872 , -0.16269737, 3.57268358, 1.30299727, -2.76138184, -0.87388793, -1.26818403]])

Some of the regression coefficients are quite small this indicates that these particular features have minimal influence on the regression analysis.[Jam+13] Considering also the computational time required for the regression, we want to try to reduce the dimensionality of our input space by feature selection.

3.1.1 Feature Selection

By reducing the input dimensionality through feature selection, we can enhance computational efficiency and potentially improve the interpretability and predictive power of the regression model.[CS14] We performed a backward feature selection with our logistic regression model. The process of backward feature selection iteratively eliminates features that contribute less to the model's performance, aiming to retain the most relevant and influential features.[Sri19] Backward feature selection was applied to check whether our assumption that we can remove features from our logistic regression model was correct or a model encompassing all features would yield to the best result. Additionally, this feature selection technique is compatible with a wide range of machine learning models, enabling its applicability across different models for result comparison and analysis. To perform the backward selection we used the SequetialFeatureSelector provided by mlxtend [3]. Figure 4.2 shows the the accuracy of our logistic regression model in relation to the number of features that were used in the generation of the model.

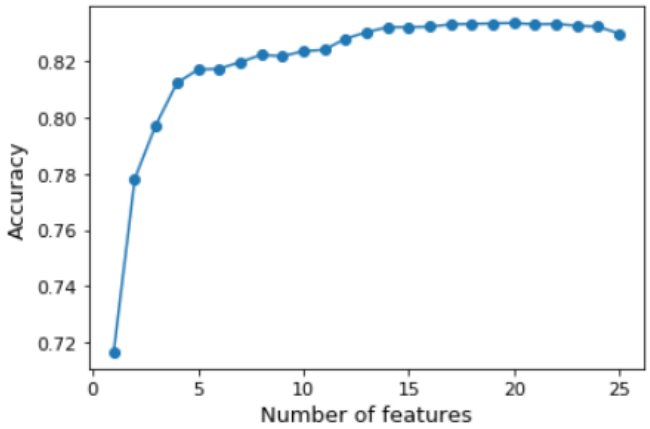


Figure 3.2: Backward feature selection development of accuracy

[3] https://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/

The accuracy of the backward feature selection demonstrates convergence after 13 features. This implies that a simplified model comprising only these 13 essential features can achieve comparable performance to a more complex model with 25 features. According to this observation, we proceeded to create a refined dataset, consisting exclusively of these 10 most significant features as identified through the feature selection process. Subsequently, we recomputed the logistic regression model with fine-tuning the hyperparameter C again to ensure optimal alignment with the updated feature set.

	fit_time	score_time	test_accuracy	test_f1_macro	test_precision_macro	test_recall_macro
0	0.075931	0.002547	0.830467	0.830029	0.833252	0.830203
1	0.031249	0.015625	0.826372	0.826274	0.826750	0.826242
2	0.031248	0.015629	0.833743	0.833707	0.833825	0.833679
3	0.031249	0.015664	0.827049	0.826526	0.830233	0.826725
4	0.031214	0.000000	0.834426	0.834101	0.836418	0.834172

Figure 3.3: Results of logistic regression with feature selection

The outcomes of the logistic regression show minimal variation to the ones before, with all accuracy measures remaining consistently around 83%. Notably, the hyperparameter C has remained unchanged, maintaining its value of 21.544. This observation implies that the excluded features did not had a substantial influence on the regularization strength, reaffirming their limited impact on the model's overall performance. In addition the generation of the *LogisticRegressionCV* model is now faster to compute. These results imply that the logistic regression model comprising only 13 features, is equally as good as a model with all 25 features thus demonstrating the efficacy of feature selection in attaining a simpler yet equally effective model.

3.2 Decision tree

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Since our dataset is balanced, we can use DT. Tree based models calculates feature importance for they need to keep the best performing features as close to the root of the tree. Constructing a decision tree involves calculating the best predictive feature. The number of samples required to populate the tree doubles for each additional level the tree grows to. Use maxdepth to control the size of the tree to prevent overfitting, that is why we use parameter tuning to find what would be the best depth parameter for our tree.

3.2.1 Parameter tuning

First run the default decision tree with all features and without any optimized hyperparameters and we are able to get the results table as shown in the 3.4. The accuracy we get is around 80%.

	fit_time	score_time	test_accuracy	test_f1_macro	test_precision_macro	test_recall_macro
0	0.197731	0.015622	0.808354	0.808178	0.809127	0.808207
1	0.234441	0.015568	0.805078	0.805078	0.805109	0.805116
2	0.187498	0.000000	0.797707	0.797705	0.797816	0.797774
3	0.187516	0.000000	0.790164	0.790150	0.790392	0.790252
4	0.226426	0.006410	0.808197	0.808065	0.808697	0.808059

Figure 3.4: Decision tree results

A complete visualisation of the tree can be viewed in appendix **??**. Clearly it is a very complex tree, and not very balanced, so we decide to do a hyperparameter tunning, expecially the depth of the tree which can give us better performance if it rightfully tuned.

We define the space of the hyperparameters: criterion = ['gini', 'entropy'] max dephts = [None,2,4, 8, 12, 16, 20]

After the parameter tuning we see that The best hyperparameters are 'criterion': 'entropy', 'max depth': 4 and the best accuracy score it provides for the training dataset is 0.8137, so 81%, improving form the default one with 1%. Below is shown the full table of results after turning. The result are improved and also the complexity of the tree is improved as well. A

	fit_time	score_time	test_accuracy	test_f1_macro	test_precision_macro	test_recall_macro
0	0.149031	0.011997	0.806716	0.806518	0.807589	0.806559
1	0.084540	0.004004	0.826372	0.826362	0.826362	0.826362
2	0.057306	0.000000	0.814906	0.814900	0.815087	0.814992
3	0.062538	0.015593	0.791803	0.791758	0.791882	0.791741
4	0.062497	0.000000	0.809016	0.808995	0.809348	0.809121

Figure 3.5: Enter Caption

decision tree with a shorter depth is typically easier to interpret and understand. It has fewer

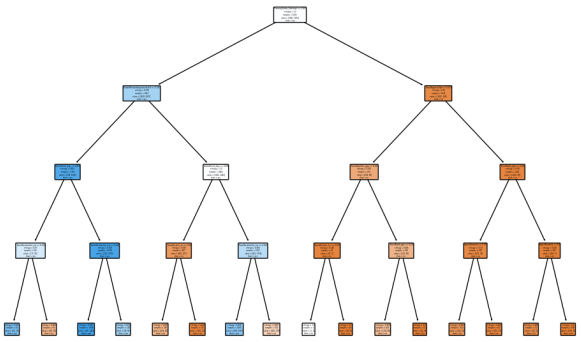


Figure 3.6: Decision tree after parameter tuning

decision rules and branches, making it more straightforward to follow and explain to others as well as reduces overfitting. They are less likely to memorize noise or irrelevant patterns in the training data, leading to improved performance on unseen examples.

3.3 Random Forest

Given the ensemble-based approach, adaptability, resilience, and interpretability of random forests, they have emerged as the favored choice for addressing the specific binary classification problem at hand. This algorithm's capability to handle both numerical and categorical target variables, coupled with its robustness against noisy and outlier instances, establishes a strong foundation for precise and dependable predictions.

- Random forests are for supervised machine learning, where there is a labeled target variable.

- Random forests can be used for solving regression (numeric target variable) and classification (categorical target variable) problems.
- Random forests are an ensemble method, meaning they combine predictions from other models.
- Each of the smaller models in the random forest ensemble is a decision tree. [Sha23]

Based this characteristics it looks like a good choice of model for us to experiment with our dataset. As in the other algorithms we utilize the Scikit-learn Random Forest [4] to perform this algorithm.

For our experiments we have chosen the parameters as below: criterion = 'entropy' , max depth = None, n estimators = 100. Below is shown the table with the results.

	fit_time	score_time	test_accuracy	test_f1_macro	test_precision_macro	test_recall_macro
0	2.114388	0.046841	0.855037	0.854924	0.855782	0.854905
1	1.973171	0.031213	0.860770	0.860702	0.861924	0.860980
2	1.790135	0.031240	0.848485	0.848465	0.848917	0.848615
3	1.807847	0.046892	0.851639	0.851634	0.851632	0.851638
4	1.788268	0.031290	0.855738	0.855724	0.855756	0.855710

Figure 3.7: Results of Random Forest

As we can see from the from the 3.7 the results presented from this algorithm are quite promising in the training. In the figure are shown the results of the default configuration of hyper parameters. In the second step we decide to tune the hyperparmenters in order to find the best combination that gives us the best accuracy.

3.3.1 Parameter tuning

In line with other models that we are experimenting, we will use the same methodology for random forest instead it seems that most practical guidance to improve RF performance is on tuning the algorithm hyperparameters, arguing that Random Forest as a tree-based method has built-in feature selection, alleviating the need to remove irrelevant features.

In the first run we used the default values of hyperparameters. The default values for the parameters controlling the size of the trees (e.g. max depth, min samples leaf, etc.) lead to fully grown and unpruned trees which can potentially be very large on some data sets. To reduce memory consumption, the complexity and size of the trees should be controlled by setting those parameter values.

The main parameters to adjust when using these methods is n_estimators and max features [5] Taking into consideration the sikit-learn. guidelines, to tune the hyperparameters specify the parameters space as below:

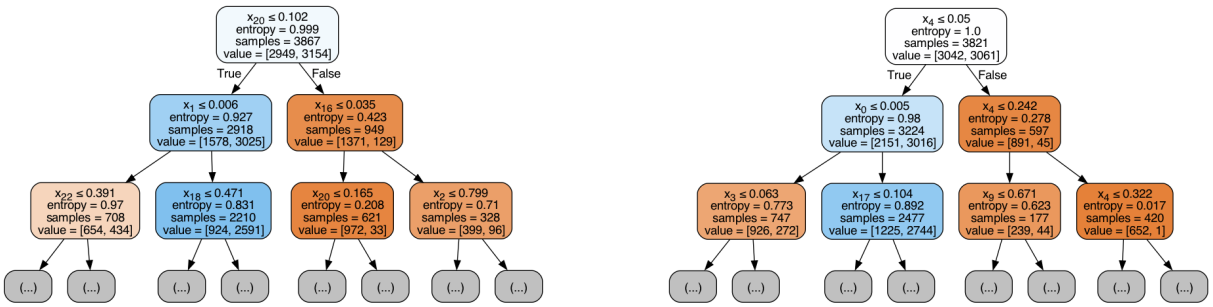
- ntrees = [100,200,300,None]
- max_depth = [50, 100,200,None]
- criterion = ['gini', 'entropy']

[4] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.htm

[5] https://scikit-learn.org/stable/modules/ensemble.html#random-forest-parameters

After performing the algorithm we are able to get the best hyperparmaeters to re-run the algorithm. The best accuracy score for the training dataset is 0.8527. The best hyperparameters are 'criterion': 'entropy', 'max depth': 50, 'n estimators': 400.

Below are shown three[6] of the tree generated by random forest.



(a) Subtree 1 (b) Subtree 2

Figure 3.8: Two Subtrees

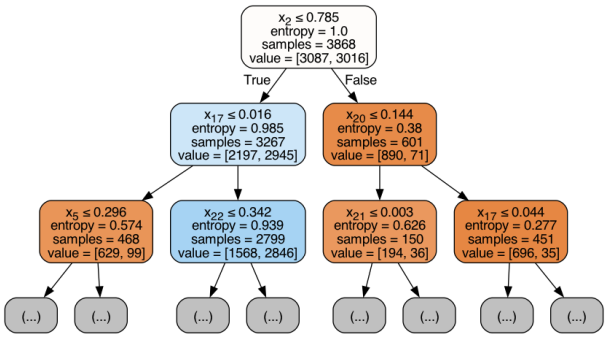


Figure 3.9: Subtree 3

In the end we have a new result table 3.10 that is actually very similar with our first but it gives us a slightly better accuracy than the one with the default parameters in 3.7.

	fit_time	score_time	test_accuracy	test_f1_macro	test_precision_macro	test_recall_macro
0	10.574680	0.125001	0.859132	0.859041	0.859733	0.859015
1	11.115564	0.109383	0.862408	0.862381	0.862988	0.862557
2	10.013502	0.125620	0.855856	0.855840	0.856254	0.855981
3	10.663059	0.156293	0.854918	0.854890	0.855009	0.854863
4	10.396069	0.124969	0.850000	0.849937	0.850302	0.849902

Figure 3.10: Random Forest after hyperparameter tunning

The results obtained after tuning are very optimistic with an accuracy around 85/

3.4 Support vector machine

A Support Vector Machine (SVM) is a supervised machine learning algorithm that in the context of classification aims to determine a decision boundary between different classes in a dataset.

[6]There are several tree generatet but for the sake of vizualization only three are shown.

It achieves this by finding an optimal hyperplane that maximizes the margin, or the distance, between the decision boundary and the closest data points from each class, known as support vectors. [SW17]

We utilize the Scikit-learn SVC model [7] to compute Support Vector Machines (SVM). This implementation offers several hyperparameters, we were particular interested in the regularization parameter C. Given that our dataset contains 25 features after preprocessing, regularization becomes crucial to prevent an overly complex model. Additionally, the choice of kernel is essential as it enables the SVM to map the input data from the original feature space to a higher-dimensional space. Considering that the previous used logistic regression model assumes linear decision boundries and the decision tree and random forest model are flexible and can capture both linear and non-linear relationships, we will test a linear kernel function and a radial basis kernel function (RBF) to determine if our data is better describeable with linear or non linear relationships. For the hyperparameter C, we will test a logarithmic scale with 6 values ranging from 10^{-2} to 10^{3}. This wide range allows us to consider both very small and very large values for regularization, covering a broad spectrum of possibilities. By doing so, we can thoroughly explore the regularization strengths and select the most appropriate value for our SVM model. As for the decision tree and random forest models we used the grid search algorithm to optimize the hyperparameters. The optimal hyperparameters for our training data are C = 1.0 and a RBF kernel meaning non linear decision boundaries are describing our data better.[SW17] Figure 3.11 shows the results achieved by training an SVM model with the determined hyperparameters, fitting it to our training dataset and evaluating its performance using cross validation.

	fit_time	score_time	test_accuracy	test_f1_macro	test_precision_macro	test_recall_macro
0	1.026461	0.412735	0.847666	0.847378	0.849733	0.847445
1	0.778905	0.306187	0.839476	0.839463	0.839475	0.839455
2	0.804011	0.345656	0.844390	0.844389	0.844401	0.844418
3	0.849653	0.337973	0.845902	0.845808	0.846391	0.845777
4	0.780386	0.369460	0.845082	0.844954	0.845808	0.844930

Figure 3.11: Results of SVM

The results obtained from the various cross-validation folds and evaluation metrics are very consistent, as they constantly hover around 84%. That the results across the different evaluation metrics are very similar means that the model is achieving a balanced performance in terms of classifying positive (Exoplanet candidate) and negative (false positive) instances.

3.4.1 Feature Selection

In line with the methodology applied for the logistic regression model feature selection was performed for SVM model to check if we could obtain similar results with less features resulting in a simpler model. For the feature selection we computed again a backward feature selection. The SVM model used for the feature selection corresponds to the one with the optimal hyperparameters determined in the previous step. Figure 3.13 shows the accuracy of our SVM model in relation to the number of features that were used in the model generation.

[7] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

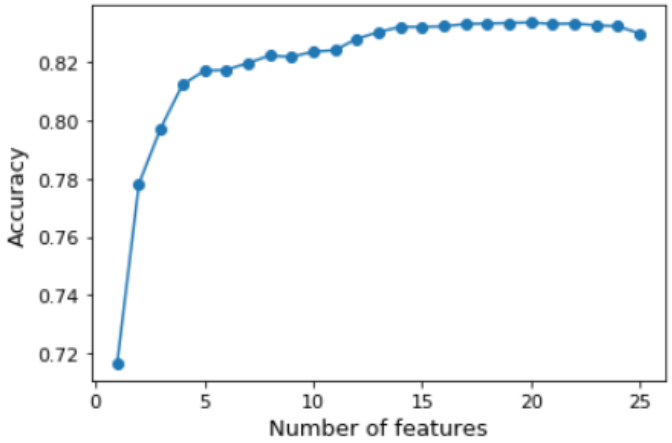


Figure 3.12: SVM Backward feature selection

The accuracy of the backward feature selection demonstrates convergence after 14 features. The following 10 features were selected by the feature selection of the logistic regression and the SVM model making them the most important features in our dataset: *OrbitalPeriod_days, TransitDuration_hr, PlanetaryRadius_Earthradii, EquilibriumTemperatureK, StellarEffectiveTemperatureK, StellarSurfaceGravity_log10(cm/s2), RA_decimaldegrees, Dec_decimaldegrees, Keplerband_mag, TransitDurationUnc_hrs*

We again proceeded with creating a refined dataset, consisting of the 14 features identified through the feature selection. Subsequently, we recomputed the SVM model with optimizing again the hyperparameters C and kernel with Grid Search to ensure optimal alignment with the updated feature set.

	fit_time	score_time	test_accuracy	test_f1_macro	test_precision_macro	test_recall_macro
0	1.639115	0.671015	0.850123	0.849760	0.852880	0.849869
1	1.214853	0.529418	0.845209	0.845167	0.845338	0.845134
2	1.012024	0.499012	0.850942	0.850942	0.851004	0.850994
3	1.276236	0.583367	0.849180	0.848947	0.850769	0.848959
4	1.056926	0.387486	0.847541	0.847360	0.848690	0.847352

Figure 3.13: Results of SVM with feature selection

The outcomes of the SVM with feature selection show minimal variation to the ones before without feature selection, with all accuracy measures remaining consistently around 84%. The hyperparameter C and the optimal kernel have remained unchanged. These results imply that the SVM model comprising only 14 features, is equally as good as a model with all 25 features thus demonstrating the efficacy of feature selection in attaining a simpler yet equally effective model.

Chapter 4

Results and Conclusion

4.1 Comparison of the Models

This project used four different machine learning models to analyze our NASA exoplanet dataset. The results for each model, including details on hyperparameter optimization and feature selection, are presented in figure 4.1.

	Model	Details	Accuracy	F1 Macro	Precision Macro	Recall Macro
2	Random Forest	Optimized hyperparameters criterion, max_depth and n_estimators	0.855480	0.855426	0.856037	0.855509
3	SVM	performed feature selection (14 features) and hyperparameter optimization (kernel,C)	0.848599	0.848435	0.849736	0.848462
0	Logistic Regression	Feature Selection (13 features), hyperparameter C optimized	0.830411	0.830127	0.832096	0.830204
1	Decision Tree	Optimized hyperparameters criterion (entropy) and max_depth (4)	0.794526	0.794457	0.794798	0.794482

Figure 4.1: Comparison of the different models

Among the tested models, the Random Forest model demonstrated the best performance across all evaluation metrics. It achieved the highest accuracy, F1 score, precision and recall meaning it yields the best results for both the exoplanet candidate and false positive class. Following Random Forest were the SVM and logistic regression model. The lowest accuracy was achivied with the decision tree model. The contrasting performance of Random Forest and the decision tree model highlights the advantages of ensemble learning. By aggregating predictions from multiple weaker learners (individual decision trees), the Random Forest model effectively mitigated the limitations associated with the decision tree model, resulting in more robust and accurate predictions.

With an accuracy of 85%, the Random Forest model achieved satisfactory results for the classification task of determining whether a Kepler object of interest (KOI) is an exoplanet candidate. Considering that the classification as a candidate assumes subsequent steps in the real-world scenario to confirm its status as an exoplanet, a 15% misclassification rate is acceptable. Further checks and analysis would likely rectify any classification errors.

In conclusion, the Random Forest model emerges as the optimal choice for our use case. It consistently outperformed other models and exhibited strong performance across all evaluated criteria and in addition it did not require to perform a feature selection.

4.2 Generalization

Generalization is a crucial aspect of the model's performance as it reflects how effectively the model can predict outcomes on unseen data. To assess generalization, we initially partitioned our data into separate training and test sets. The test set, excluded from the model generation process, serves as the indicator of the model's ability to generalize. Ideally, we aim for a similar performance between the test set and the training set, signifying strong generalization capabilities.[Stö19] Upon evaluating the performance of our chosen Random Forest model on the test

data set, we obtained the following results

	precision	recall	f1-score	support
0	0.85	0.84	0.84	1482
1	0.84	0.85	0.85	1525
accuracy			0.85	3007
macro avg	0.85	0.85	0.85	3007
weighted avg	0.85	0.85	0.85	3007

Figure 4.2: Generalization of the random forest model

The classification metrics hover around 85%. Notably, this aligns exactly with the performance achieved during the cross-validation process with the training data set. This similarity indicates that our random forest model has a robust generalization ability, instilling confidence in its capacity to yield accurate predictions on unseen data.

4.3 Possible Extensions

Given that random forest the best model for our dataset is an ensemble model we could further explore the approach of ensemble models. Especially as we have already four different models trained to our dataset we could combine them using a voting classifier and evaluate if that would further enhance the performance. Additionally, to explore alternative modeling approaches and gain new insights from the dataset, we can consider employing neural networks. Neural networks offer a different approach compared to traditional classification algorithms. They have the potential to capture intricate patterns and relationships in the data through their complex architectures, thus presenting an opportunity for further performance improvement and valuable discoveries.[Zha00]

By incorporating ensemble modeling and exploring neural networks as alternative methodologies, we could broaden our analytical scope and potentially uncover additional nuances within the dataset and increase the accuracy of our predictions.

4.4 Conclusion

This report outlines our efforts to develop an accurate model for predicting the candidacy of Kepler objects of interest as exoplanets. After preprocessing the data and evaluating multiple classification algorithms, we found that Random Forest performed the best, achieving approximately 85% accuracy and excellent generalization performance. The model provides valuable insights into exoplanetary systems and demonstrates the potential of machine learning in advancing our understanding of celestial phenomena.

Appendix A

Table A.1: Features removed from the dataset

Feature	Reason
KepID	Kepler Identification number is the unique ID of every row in our dataset and will therefore add no value to our analysis.
KOIName	A number used to identify and track a Kepler Object of Interest (KOI). It can be viewed as the name of the object, but is also unique for every row in our dataset and therefore adds no value to our analysis.
KeplerName	Kepler number name in the form "Kepler-N," plus a lowercase letter. It is easier to remember than the KOIName but has otherwise the same function and is therefore not useful for analysis purposes.
ExoplanetArchiveDisposition	As explained in XX, we will not try to predict the archive disposition. Therefore, we need to remove it from our dataset as it will be highly correlated with our target variable, the Kepler Disposition.
TCEDeliver	TCE (Threshold Crossing Event) name of the data delivery federated to the KOI (Kepler object of interest). It is also a name and does not provide data coming directly from the Kepler telescope.
TCEPlanetNumber	TCE (Threshold Crossing Event) planet number is also federated to the KOI, so it is external data that does not come from the Kepler telescope itself.
DispositionUsingKeplerData	Our target variable that was transformed in the previous step to a binary variable called Exoplanet candidate. Hence, we can drop the string representation of it.
DispositionUsingKeplerData	Our target variable that was transformed in the previous step to a binary variable called Exoplanet candidate. Hence, we can drop the string representation of it.
NotTransit-LikeFalsePositiveFlag koi_fpflag_ss CentroidOffsetFalsePositiveFlag EphemerisMatchIndicates- ContaminationFalsePositiveFlag	Flags that are set by external systems or programs. Since we were not able to understand exactly on what basis the flags are set, based on the description provided by CalTech, and correlations with our other features could not be excluded, we decided not to include the flags in the analysis.

Appendix B

Image of the whole decision tree is shown on the next page

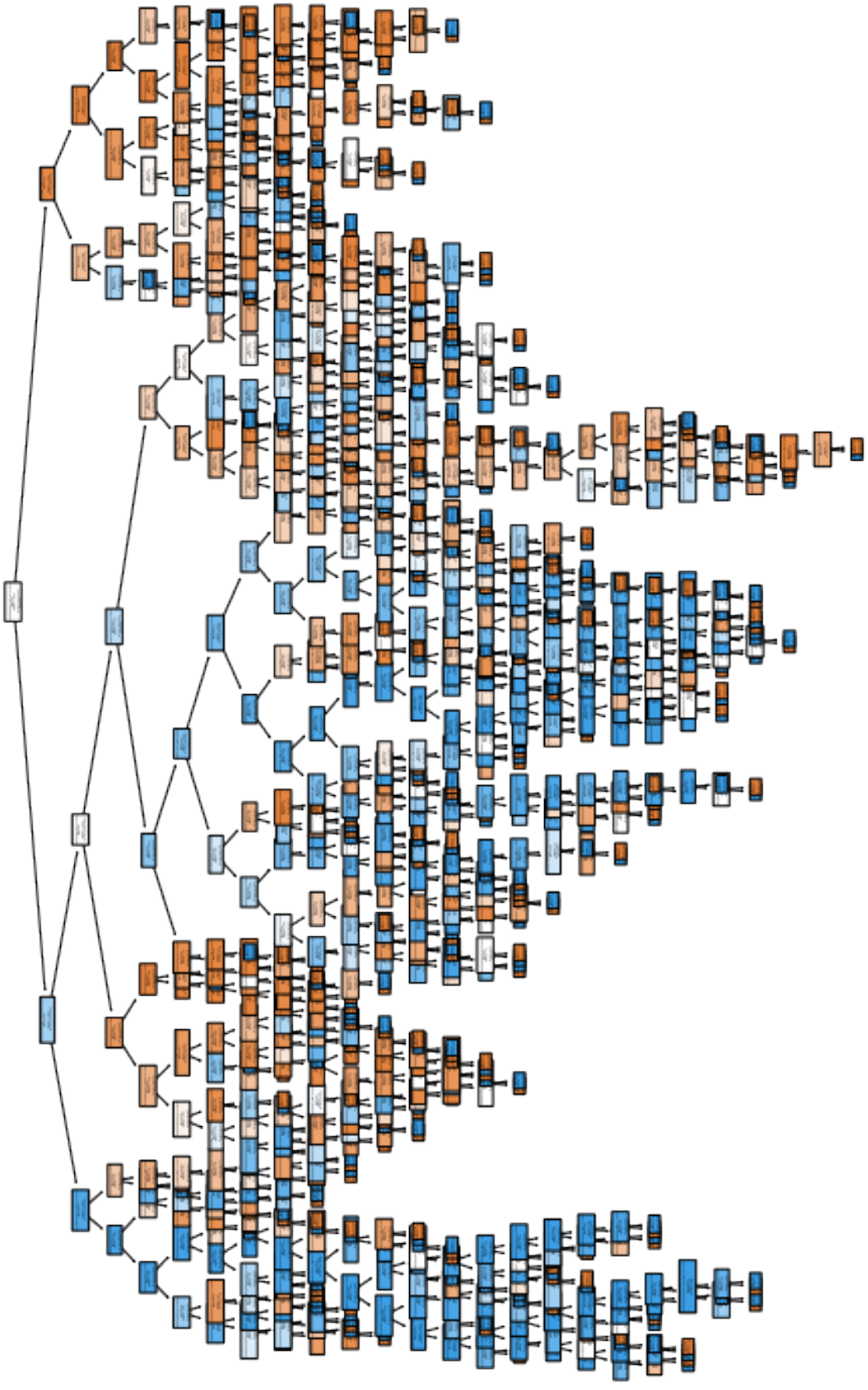


Figure B.1: Decision tree visualization

Bibliography

[CS14] Girish Chandrashekar and Ferat Sahin. "A survey on feature selection methods". In: *Computers and Electrical Engineering* 40 (2014), pp. 16–28. URL: https://www-sciencedirect-com.recursos.biblioteca.upc.edu/science/article/pii/S0045790613003066.

[Cho20] Jason Chong. *What is Feature Scaling & Why is it Important in Machine Learning?* Accessed on 03.06.2023. 2020. URL: https://towardsdatascience.com/what-is-feature-scaling-why-is-it-important-in-machine-learning-2854ae877048.

[Jam+13] Gareth James et al. *An Introduction to Statistical Learning*. 2013, pp. 204–230.

[Nai22] Aashish Nair. *Standardization vs Normalization*. Accessed on 03.06.2023. 2022. URL: https://towardsdatascience.com/standardization-vs-normalization-dc81f23085e3.

[Pin21] Koo Shung Ping. *Accuracy, Precision, Recall or F1?* Accessed on 03.06.2023. 2021. URL: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9.

[PV18] Jason Poulos and Rafael Valle. "Missing Data Imputation for Supervised Learning". In: *Applied artificial intelligence* 32 (2018), pp. 186–196. URL: https://www.tandfonline.com/doi/epdf/10.1080/08839514.2018.1448143?%20needAccess=true&role=button.

[SW17] Claude Sammut and Geoffrey Webb. *Encyclopedia of Machine Learning and Data Mining*. 2017, 780 - 781 and 941 -946 and 1049 –1054.

[Sha23] Adam Shafi. *Random Forest Classification with Scikit-Learn*. Accessed on 03.06.2023. 2023. URL: https://www.datacamp.com/tutorial/random-forests-classifier-python.

[Sri19] Sunny Srinidhi. *Backward Elimination for Feature Selection in Machine Learning*. Accessed on 03.06.2023. 2019. URL: https://towardsdatascience.com/backward-elimination-for-feature-selection-in-machine-learning-c6a3a8f8cef4.

[Stö19] Andreas Stöffelbauer. *Understanding Generalization, Regularization, Overfitting, Bias, and Variance in Machine Learning*. Accessed on 03.06.2023. 2019. URL: https://towardsdatascience.com/generalization-regularization-overfitting-bias-and-variance-in-machine-learning-aa942886b870.

[Zha00] Guoqiang Peter Zhang. "Neural Networks for Classification: A Survey". In: *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS* 30 (2000). URL: https://www.math.pku.edu.cn/teachers/ganr/course/pr2010/Ref/zhang00_nn_survey.pdf.