Final Project Report
Warfarin Dose Prediction using Machine Learning

Abstract: Warfarin is a prescription drug which interferes with normal blood coagulation. Prescribing an accurate dosage of Warfarin is of great importance to avoid serious complications. In this project, I developed several machine learning models aiming to aid clinical professionals prescribe the correct dosage based on the most important patient's features. Our models achieved a high degree of accuracy, between 75 and 79% depending on the method used. A Gradio interface was also developed to facilitate the use of the models.

Nexhi Sula

December 16, 2022

Introduction:

Warfarin is a prescription medication that interferes with normal blood clotting or coagulation. It is prescribed to approximately 2 million Americans annually who are at a high risk of developing harmful blood clots, which can cause serious medical problems such as strokes, heart attacks, or pulmonary embolisms. Prescribing an accurate dosage of warfarin is of great importance, as prescribing too high of a dose can lead to serious complications, including excessive internal bleeding (UpToDate, 2022). HoIver, the population taking warfarin is diverse, as one may be at risk of blood clots for a multitude of reasons. As a prescriber, it can be difficult to discern the influence certain clinical, demographic, and pharmacogenetic patient characteristics should have when prescribing.

For over a decade, doctors and scientists have been developing and improving tools that can help them to prescribe the correct dosage. Dr. Brian Gage was the first to do so, and in 2008, he applied 2 linear regression models to predict the therapeutic dose of warfarin (Gage et al., 2008). HoIver, his initial dataset was limited to 1015 patients. This prompted the IWPC (International Warfarin Pharmacogenetics Consortium) to compile data on 5052 patients and propose a clinical and a pharmacogenetic linear regression model (IWPC, 2009). Since then, researchers and clinicians have developed several variations of these models, some being race and age-specific models, some incorporating classification along with regression, one using a stacked ensemble approach, and many others (Cosgun et al., 2001; Sharabiani et al., 2015; Nowak-Gottl et al., 2010; Ma et al., 2018). This topic is still under much discussion regarding the incorporation of certain features and their relationships to one another, and the scientific community is open to new and improved models.

I was excited by the challenge this problem presents, and based on the IWPC dataset, created our own model to guide patients to the correct dose of warfarin. My model groups patients into one of two classes: those who require >30 mg warfarin per Iek and those who require ≤ 30 mg/wk. I first carefully pre-processed the dataset and incorporated the steps into a streamlined pipeline. I then tested 5 models–logistic regression, support vector machine, decision tree, random forest, and neural network. I evaluated these models using accuracy, precision, recall and F1. I used GridSearchCV with the training set on the first four models to identify the optimal parameters, and optimization of the neural network using the training and validation sets to determine the best number of layers and neurons with BCE as our loss function. Ultimately, the best model was our logistic regression model which performed with 79% accuracy on the test set. HoIver, all models Ire deployed on Gradio, so that the user may select the desired model.

Methods: Pre-processing

Loading Data and Initial Feature Reduction:

The "IWPC Data Set" was downloaded from the https://www.pharmgkb.org/downloads website found under the heading "International Warfarin Pharmacogenetics Consortium (IWPC)". It was loaded into my jupyter notebook as a pandas DataFrame using `pandas.read_csv`. The dataset originally contained 70 features, including our label 'Therapeutic Dose of Warfarin'. The features Ire reduced to the following: 'Gender','Race (Reported)', 'Age', 'Height (cm)', weight (kg)', 'Diabetes', 'Simvastatin (Zocor)', 'Amiodarone (Cordarone)', 'Target INR', 'INR on Reported Therapeutic Dose of Warfarin', 'Cyp2C9 genotypes','VKORC1 genotype: -1639 G>A (3673); chr16:31015190; rs9923231; C/T', 'Therapeutic Dose of Warfarin'. I visualized it using matplotlib functions as Ill as pandas DataFrame functions.

Dividing into training and test sets and Initial Pre-Processing:

I set aside a validation set of 5% immediately using sklearn's train_test_split. This was done to ensure that our pipeline will be able to correctly process untouched new data. I then dropped any rows with a null label (i.e. Nan in the column 'Therapeutic Dose of Warfarin') The remaining non-validation set contained 5246 entries. Before separating the data into training and test sets, I took into consideration that weight is often an important determinant of medication dosage. Therefore, I utilized stratified sampling to ensure that the test set was representative of the various categories of weight in the whole dataset (Geron, Ch2, 2019). In order to perform stratified sampling based on the "weight (kg)" category, I had to first eliminate any null weight values (Nan). To address this, I chose to fill in null weight values with the median by gender. To accomplish this, I had to address null gender entries, of which there were only 4. Because of this low number, I chose to drop these entries altogether. However, for future cases in which gender is Nan, I will replace it with the mode (male). After performing this minimal pre-processing on the combined test and train dataset, I performed an 80:20 split using StratifiedShuffleSplit from sklearn.

Addressing Features One by One

weight (kg): Our program addresses any incoming null weight values with replacement using the median weight by gender, as it did on the dataset prior to the stratified split. Notably, the weight class contains outliers, as visualized in Supp. Fig 1. These can distort the distribution and decrease the statistical power of our data. However, because they appear to represent natural variation in the population, I chose not to perform any additional pre-processing on these outliers (Frost, 2022). I performed Standard Scaling normalization for weight (kg).

Height (cm): Similar to the correlation between weight and gender, height is also correlated with gender. The median height among males in the training dataset was 173.74 cm, while the median height for females in the dataset was 160.02 cm. Therefore, I replaced null height values with the median by gender. I performed Standard Scaler normalization for Height (cm).

Race (Reported): The race feature was a bit tricky. First, some groups Ire redundant–i.e. “Other” and “other”. To address this, I capitalized all entries belonging to the race feature as part of the pre-processing. Similarly, there Ire separate groups for “Black”, “African-American”, and “Black or African American.” This redundancy is explicit, as there is already a group “Black or African American” encompassing both nomenclatures. There were other groups that appeared redundant–i.e. “White” and “Caucasian”. However, I did not take liberties in combining any groups unless there was already a group doing so.

There Ire 371/4196 null entries in the training set. To avoid oversimplifying the data by simply assigning the mode race (“White”) to all entries with null race, I pre-process the data to assign any entries with Nan for race into a category titled: "UNSPECIFIED" as it was evidently not reported. These collective changes can be visualized in Supplementary Figure 2.

Regarding the transformation of race from a text feature to a numerical one, I chose to use **One-Hot Encoding**, as race is not an ordered variable. I noticed that most entries belonged to one of seven racial categories. However, several entries belonged to a unique racial group, or a racial group with less than 10 entries. Therefore, our analysis of the training set does not capture the feature variation among the entire population, as there are unique racial groups represented in the testing and validation set that are not in the training set. Therefore, I prepared our program to encounter new categories for race. To prevent the program from erroring, I specified “`handle_unknown='ignore'`” when initializing the OneHot Encoder. This does the following: “When an unknown category is encountered during transform, the resulting one-hot encoded columns for this feature will be all zeros.”

Age

Age is represented in this dataset by categorical brackets of 10 years, rather than as a scalar value. 23/4196 Ire null values in the training set, with 70-79 as the most Ill represented age bracket by a small margin. Due to the small number of entries with a null value, I decided to assign these 23 entries with the mode, and to do the same with future entries. To transform the feature from text to numerical, I applied OrdinalEncoder, because age is an ordered variable.

Diabetes, Simvastatin, and Amiodarone

Diabetes, Simvastatin and Amiodarone are each binary categorical variables represented by 0 or 1. The training set contained 1815/4196 null entries for diabetes, 1336/4196 for Simvastatin, and 1142/4196 for Amiodarone . The number of individuals in the training set without diabetes is much greater than those that have it. Similarly, the number of individuals not taking the drug is also much greater than those taking it, for both drugs. Therefore, I replaced any null values in each of these three columns with a 0, indicating that the individual does not have diabetes or indicating that the individual is not taking the drug. I transform new data with the mode from the training set as Ill.

Cyp2C9 genotypes

The Cyp2C9 genotypes feature showed 72/4196 null values in the training set, and 3108 entries with the *1/*1 genotype, making it the most common by far out of 11 shown. Based on the association between genotype and race, it may have been more appropriate to fill null values based on the mode for each race (Limbdi et al. 2015). However, to avoid overcomplicating the pre-processing pipeline, I simply replaced any null values with the training set mode (*1/*1) and transform new data with this same mode. I then applied the same OneHotEncoder to this feature.

VKORC1 genotype

The VKORC1 genotype feature showed 1232/4196 null values in the training set and a fairly even distribution among three genotypes–A/A, A/G, G/G. Because of the large number of null values and lack of a single most prevalent genotype, I created a new category "Unknown" for the missing values. I then applied the same OneHotEncoder to this feature. New data will be transformed this same way.

Target INR and INR on Reported Therapeutic Dose of Warfarin

Analysis of the combined test and training dataset revealed that the Target INR feature contained 4055/5246 null values. Due to the lack of data for this feature, and because a similar feature was present (INR on Reported Therapeutic Dose of Warfarin) with far fewer missing values (530/5246), I decided to drop Target INR as a feature. I incorporate this as part of the pre-processing pipeline. For INR on Reported Therapeutic Dose of Warfarin, I replace the missing values with the median of the training set (using `SimpleImputer(strategy='median')` and apply StandardScaler normalization.

Combined Attribute Features and Multicollinearity

As the textbook mentions in Chapter 2 (pages 58-62), sometimes features can be combined to create a more informative combined attribute. In the study conducted in 2008 by Gage et al. assessing predictors for warfarin dose, they utilize body surface area (BSA) instead of weight and height when predicting the correct dosage of warfarin. BSA is a common metric used for the calculation of medication dosages (medicinnet). Therefore, I found the DuBois and DuBois formula for BSA from UpToDate and added the BSA feature to our dataset. I also calculated BMI and added this feature. Knowing that the addition of these features may create additional collinear features, I looked at correlations among variables and found that BMI, BSA, Height, and weight, are all correlated with one another (Supp. Fig 3). To prevent introducing additional multicollinear variables into our dataset, I chose not to include BMI and adjusted the pre-processing pipeline to easily eliminate the weight variable. However, since I do not need to understand the role of each independent variable to successfully build the model, I did not address multicollinearity beyond what is described. For the BSA feature, I performed StandardScaling normalization.

Pre-processing pipeline

To incorporate these various pre-processing steps into a pipeline for ease of application with new data, I looked to the jupyter notebook associated with Chapter 2 of the textbook as a reference (see https://github.com/ageron/handson-ml2). Based on the examples provided, I created three of our own transformer classes–one for filling in null categorical values (`CatTransformer`), one for filling in null weight and height with the gender dependent median (`GenderTransformer`), and one for adding a new attribute, should I choose to do so (`CombinedAttributesAdder`). These transformers inherited BaseEstimator and TransformerMixin from sklearn. These classes Ire initialized by their associated pipeline (`cat_pipeline, gender_pipeline, num_pipeline`), respectively. A fourth pipeline–`scale_encode_pipeline`–was initialized as Ill. Notably this pipeline utilizes sklearn's ColumnTransformer which applies transformers to columns of an array or pandas DataFrame. Therefore, the `scale_encode_pipeline` sends categorical columns through OridnalEncoder or OneHotEncoder, and sends non-categorical columns through `num_pipeline` for imputing with the median and scaling with StandardScaler. ColumnTransformer then puts all the pre-processed data together and outputs a numpy array.

These pipelines are all integrated into a basic function named `full_preprocess_function` which can be visualized in Supp. Fig 4.

Methods: Building different ML models

For this project I decided to do a binary classification for the Warfarin dose that a person should take based on different features.

So, before building different ML models and getting the predictions, I converted the label values ('Therapeutic Dose of Warfarin') into binary classes. The first class contains patients who require doses of >30 mg/wk (high required dose (HRD)) and the second class contains the patients who need doses of ≤30 mg/wk (low required dose (LRD)).

After defining both classes for each dataset (training and testing dataset) I built the ML models which are:

1. LOGISTIC REGRESSION MODEL

Logistic regression can be used for binary classification because it estimates the probability that one instance belongs to a class or not. So, by using a probability threshold e.g 50%, it classifies the instances in positive class (1) if the probability is greater than 50 %. otherwise, the instances will be classified in negative class (0). So, this model works in the same way as the Linear Regression but instead of outputting the result, it outputs the logistic of the result.

2. SUPPORT VECTOR MACHINE

The main goal of Support Vector Machines is to fit the widest possible “street” between the classes. So, I need to have a large margin between the decision boundary which separates the classes and the training instances. The objective of SVM to find the optimal classifier is because the other linear classifiers might separate linear datasets in the correct way, but the decision boundary is so close to the training instances so that these models will probably not perform as Ill on new instances. That’s why SVM tries to find the widest possible "street" between the classes.

3. DECISION TREE MODEL

The decision tree model can be used for both classification and regression in machine learning. As the name of the algorithm, it uses a tree model to make decisions. The objective is creating a model which will predict the value of a target variable by learning simple decision rules inferred from the data features. The reason why this model is widely used in ML is because decision trees are easy to understand and interpret. It doesn’t require too much data preparation and can handle both numerical and categorical data.

4. RANDOM FOREST MODEL

Random Forest model is a supervised machine learning algorithm which is used for classification, regression based on the decision trees. So, it is a set of multiple Decision Trees combined randomly. The random forest models make predictions by collecting the prediction from each tree and then make the final predictions. Some of the most important parameters of Random Forest are: n_estimators, max_depth, max_features, max_leaf_nodes.

4. NEURAL NETWORK MODEL

Artificial Neural Networks are machine learning algorithms which can be used for classification, or regression. They are composed of node layers: input and output layers, as Ill as hidden layers between them.

ANN models make predictions by making a forward pass of the different inputs by multiplying by the weight and biases of each neuron. Each neuron uses an activation function to come up with a final value. The input layer needs as many neurons as inputs to the model and the output layer has as many output neurons as outputs there are (1 for binary classification). The most important parameters are the activation functions of each neuron, the number of hidden layers and the number of neurons per layer.

Results:

The first step was training all models in the training dataset and then getting the predictions using the training dataset. For now, the test set will remain untouched until I perform the validation and choose the best models using the hyperparameter tuning.

For evaluating the performance of each classifier, I have used these metrics:

· **Accuracy→** is the fraction of correct predictions.

· **Precision→** how many of the values that I predicted as positive are true positive

· **Recall→** how many of the relevant values are predicted as positive,

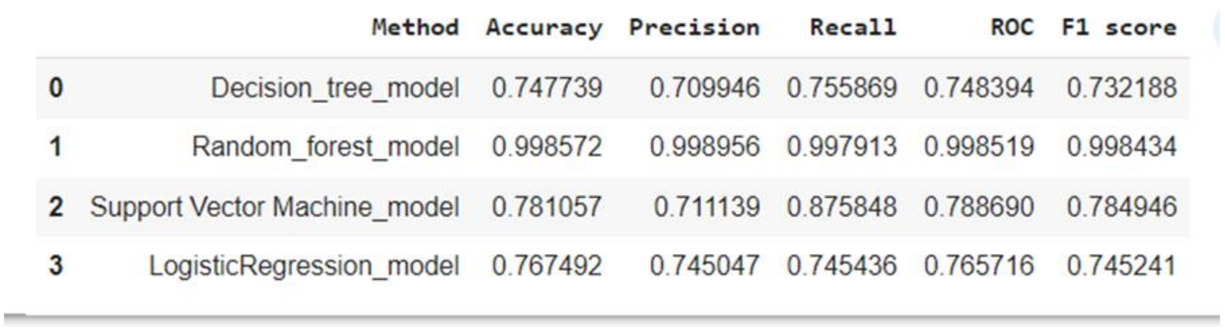
· **F1 score→** combines the precision and recall giving a single score and is defined to be the harmonic mean of the precision and recall.

F1 = 2 * (precision * recall) / (precision + recall)

· **ROC (Receiver operating characteristic) →** shows the performance of a binary classifier. The classifier with a good performance will be the one where the ROC curve is much closer to the top-left corner.

Below are shown the metrics results for every classifier on the training dataset:

Table 1: Classifier Metrics on the training dataset



	Method	Accuracy	Precision	Recall	ROC	F1 score
0	Decision_tree_model	0.747739	0.709946	0.755869	0.748394	0.732188
1	Random_forest_model	0.998572	0.998956	0.997913	0.998519	0.998434
2	Support Vector Machine_model	0.781057	0.711139	0.875848	0.788690	0.784946
3	LogisticRegression_model	0.767492	0.745047	0.745436	0.765716	0.745241

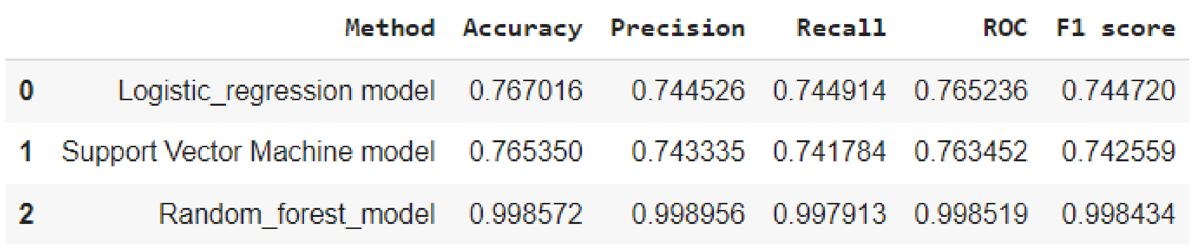
In order to choose the best model for our project I used the Cross-Validation method dividing the dataset in 10 subsets and then using the GridSearchCV to find the best combination of the hyperparameter values for each classifier. After finding the best parameters for each, I built the models using these parameters and got their predictions.

Table 2: Best parameters for each classifier

Classifiers	Best parameters calculated form Grid Search
Logistic_regression model	Best penalty: l2 Best C: 1
Support Vector Machine model	Best kernel: linear Best C: 0.1 Best gamma: 0.001
Random_forest_model	Best n_estimators: 200 Best max_features: auto
Neural Network model	Best number of layers: 4 Best number of neurons: 100

Below are shown the metrics results from every optimized classifier from cross-validation on the training set:

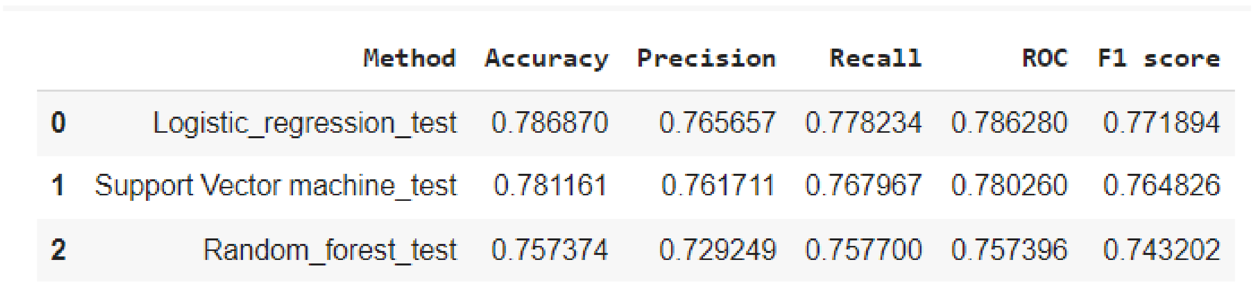
Table 3: Optimized Classifier Metrics on the training dataset



	Method	Accuracy	Precision	Recall	ROC	F1 score
0	Logistic_regression model	0.767016	0.744526	0.744914	0.765236	0.744720
1	Support Vector Machine model	0.765350	0.743335	0.741784	0.763452	0.742559
2	Random_forest_model	0.998572	0.998956	0.997913	0.998519	0.998434

Now that I have the best models with their best parameters, the last step is to evaluate each of the models in the test dataset and get the results. As mentioned before, this dataset is not seen from our models.

Table 4: Optimized Classifier Metrics on the test dataset



	Method	Accuracy	Precision	Recall	ROC	F1 score
0	Logistic_regression_test	0.786870	0.765657	0.778234	0.786280	0.771894
1	Support Vector machine_test	0.781161	0.761711	0.767967	0.780260	0.764826
2	Random_forest_test	0.757374	0.729249	0.757700	0.757396	0.743202

Another method of comparing all the models for the best performance is the ROC curve. Below I have drawn the ROC curve for all three models for validation dataset and test dataset.

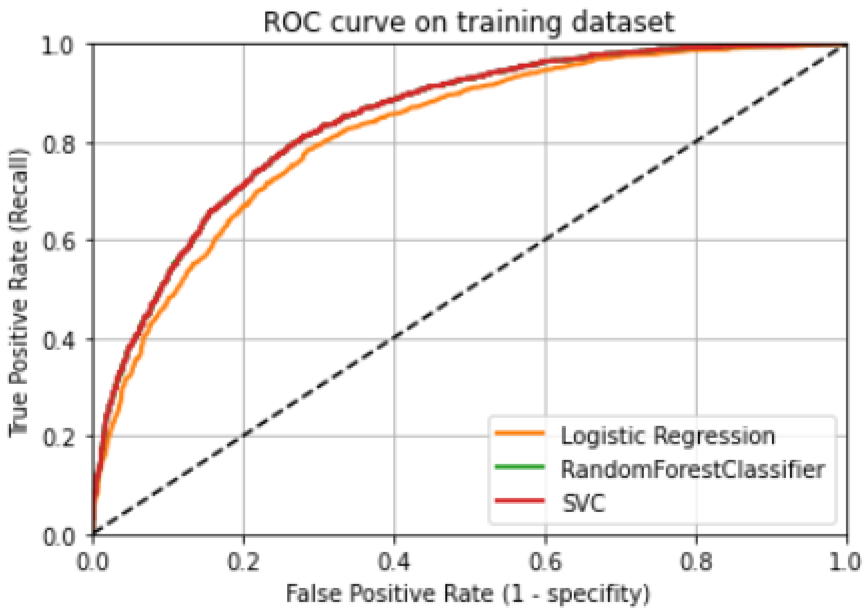


Figure 1: Optimized Classifier ROC curve on the training dataset

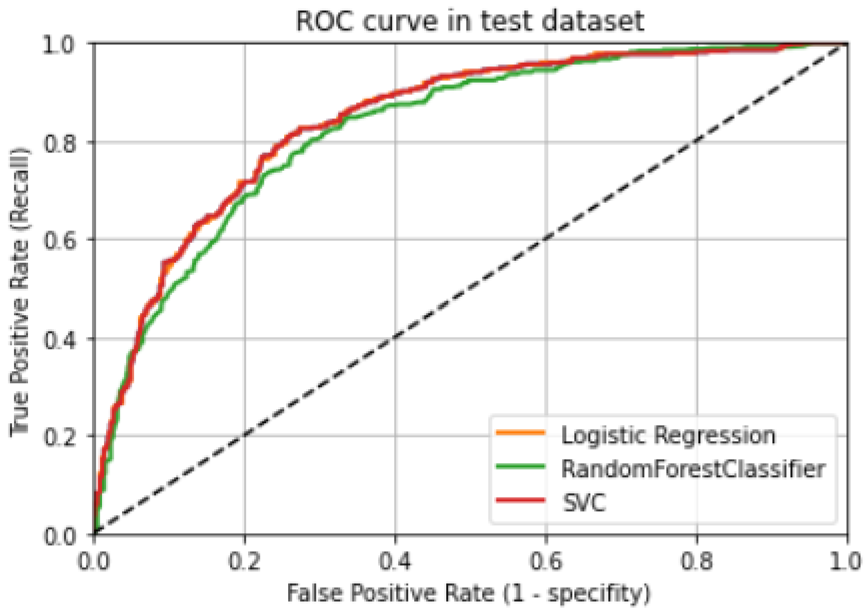
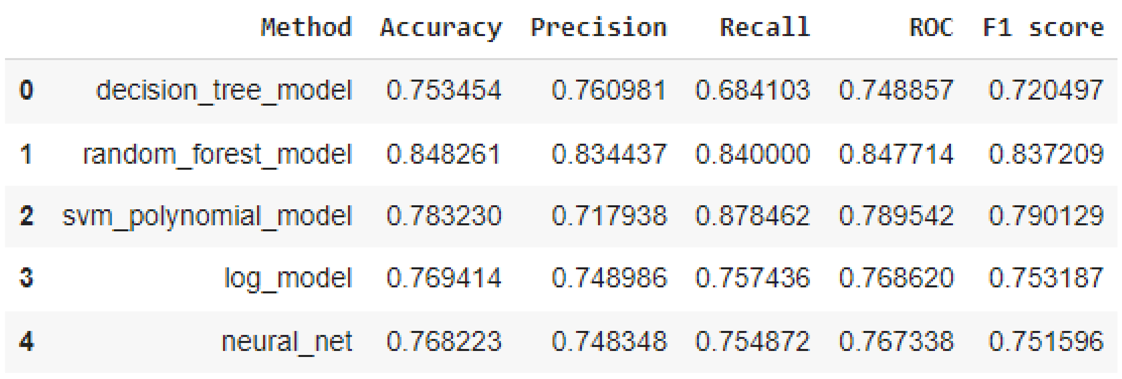


Figure 2: Optimized Classifier ROC curve on the test dataset

The Neural Network Model was trained, optimized, and its performance was measured in a different way, since the sklearn tools do not directly work with Tensorflow/Keras ANNs. All the neural networks created use ReLU activation for the hidden layer neurons and Sigmoid activation for the output layer (Sigmoid worked significantly better). Binary cross-entropy is used as the loss function, and the output layer contains only one neuron, since I am performing binary classification.

At first, an arbitrary neural network (3 layers, 1000 neurons/hidden layer) was trained. The following metrics Ire obtained when compared to the other models:

Table 5: Neural Network Classifier Metrics on the training dataset

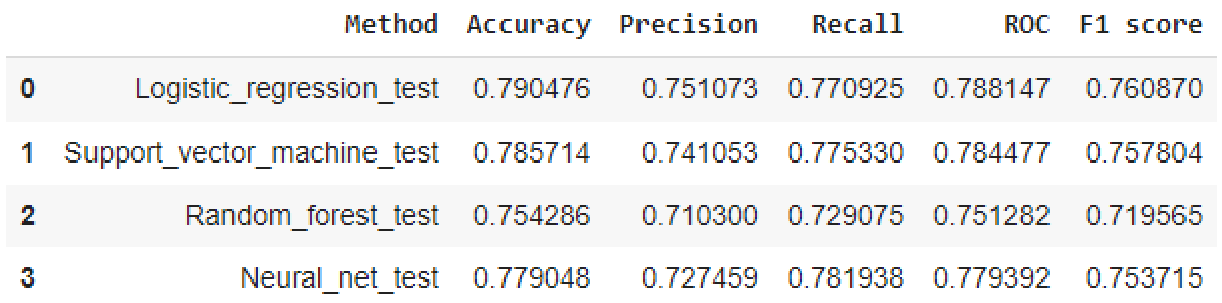


	Method	Accuracy	Precision	Recall	ROC	F1 score
0	decision_tree_model	0.753454	0.760981	0.684103	0.748857	0.720497
1	random_forest_model	0.848261	0.834437	0.840000	0.847714	0.837209
2	svm_polynomial_model	0.783230	0.717938	0.878462	0.789542	0.790129
3	log_model	0.769414	0.748986	0.757436	0.768620	0.753187
4	neural_net	0.768223	0.748348	0.754872	0.767338	0.751596

In order to optimize the ANN parameters a cross-validation loop was created. Different layer and neuron number combinations Ire tested through 30 epochs of training. Early stopping was implemented in case the validation loss stagnated to prevent overfitting.

After running the optimization algorithm, a Neural Network of 4 layers and 100 neurons per layer was selected as the best model, by comparing the best accuracy of the model in the validation dataset. Finally, the Neural Network was retrained and tested on the test set. A 15% dropout rate was defined as it offered the best performance. The final metrics Ire the compared to the other models in the following table:

Table 6: Neural Network Classifier Metrics on the test dataset



	Method	Accuracy	Precision	Recall	ROC	F1 score
0	Logistic_regression_test	0.790476	0.751073	0.770925	0.788147	0.760870
1	Support_vector_machine_test	0.785714	0.741053	0.775330	0.784477	0.757804
2	Random_forest_test	0.754286	0.710300	0.729075	0.751282	0.719565
3	Neural_net_test	0.779048	0.727459	0.781938	0.779392	0.753715

Conclusions

In the end, the best overall model was determined to be the logistic regression model, with higher scores in almost all performance metrics. This result was somewhat surprising since it is the simpler model. The threshold for the neural network was selected to be 0.5. Proper testing of different threshold values may lead to better precision/recall balance, and maybe a higher F1 score. It is up to medical professionals to determine whether a higher precision or recall is preferred for the dosage of Warfarin, since both too high or too low dosage may lead to serious complications.

With a best accuracy score of 79%, I think the goal of the model is overall achieved.

The weakest part of my approach is the fact that I performed binary classification instead of regression. While binary classification may be useful as a first approximation, it might not be precise enough for something like dosage, since a person who needs 30 mg and another who needs 90mg will be put into the same category, and reversing the doses might lead to serious consequences.

In the future, a regression model shall be created to better achieve correct dosage. For that, it is likely that more complex models such as a Neural Network will get better results than simpler models like Logistic Regression.

References

1. Aurélien Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems. Ch.2. Vol. Second edition. Sebastopol, CA: O'Reilly Media, 2019. https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=2245240&site=eds-live.
2. Cosgun, E., Limdi, N. A., & Duarte, C. W. (2011). High-dimensional pharmacogenetic prediction of a continuous trait using machine learning techniques with application to warfarin dose prediction in African Americans. Bioinformatics, 27(10), 1384–1389. https://doi.org/10.1093/bioinformatics/btr159
3. Frost, Jim. Guidelines for Removing and Handling Outliers in Data. Statistics by Jim, 2022, https://statisticsbyjim.com/basics/remove-outliers/#comments
4. Gage, B. F., Eby, C., Johnson, J. A., Deych, E., Rieder, M. J., Ridker, P. M., Milligan, P. E., Grice, G., Lenzini, P., Rettie, A. E., Aquilante, C. L., Grosso, L., Marsh, S., Langaee, T., Farnett, L. E., Voora, D., Veenstra, D. L., Glynn, R. J., Barrett, A., & McLeod, H. L. (2008). Use of pharmacogenetic and clinical factors to predict the therapeutic dose of warfarin. Clinical Pharmacology and Therapeutics, 84(3), 326–331. https://doi.org/10.1038/clpt.2008.10

5. IWPC. (2009). Estimation of the Warfarin Dose with Clinical and Pharmacogenetic Data. New England Journal of Medicine, 360(8), 753–764. https://doi.org/10.1056/nejmoa0809329
6. Limdi, Nita A et al. "Race influences warfarin dose changes associated with genetic factors." Blood vol. 126,4 (2015): 539-45. doi:10.1182/blood-2015-02-627042
7. Ma, Z., Wang, P., Gao, Z., Wang, R., & Khalighi, K. (2018). Ensemble of machine learning algorithms using the stacked generalization approach to estimate the warfarin dose. PLoS ONE, 13(10), 1–12. https://doi.org/10.1371/journal.pone.0205872
8. Nowak-Göttl, U., Dietrich, K., Schaffranek, D., Eldin, N. S., Yasui, Y., Geisen, C., & Mitchell, L. G. (2010). In pediatric patients, age has more impact on dosing of vitamin K antagonists than VKORC1 or CYP2C9 genotypes. Blood, 116(26), 6101–6105. https://doi.org/10.1182/blood-2010-05-283861
9. Sharabiani, Ashkan et al. "Revisiting Warfarin Dosing Using Machine Learning Techniques." Computational and mathematical methods in medicine vol. 2015 (2015): 560108. doi:10.1155/2015/560108
10. Stoppler, Melissa. "Medical Definition of Body surface area." MedicineNet. 2021, https://www.medicinenet.com/body_surface_area/definition.htm

Supplementary Material:

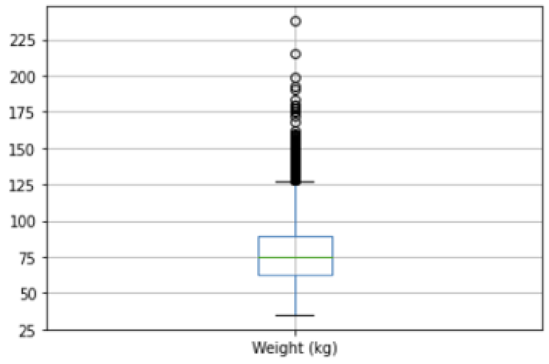


Fig 1: Distribution of un-preprocessed "Weight (kg)" features in training set

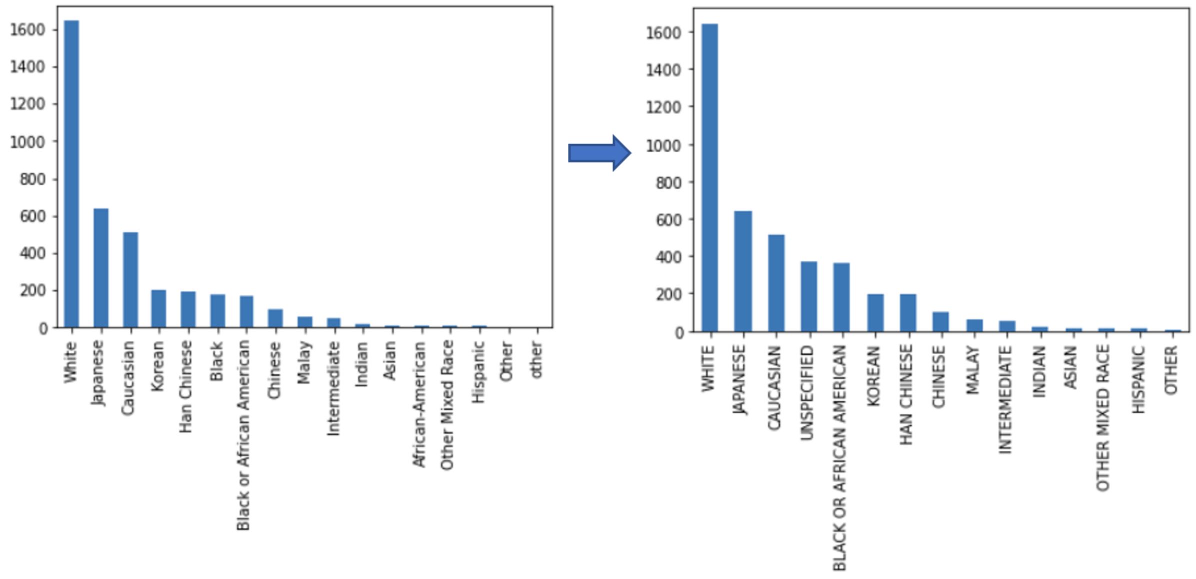


Fig 2: distribution plot on training set before and after pre-processing on 'Race (Reported)' feature

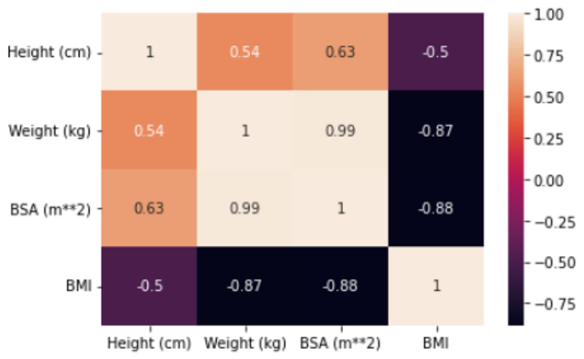
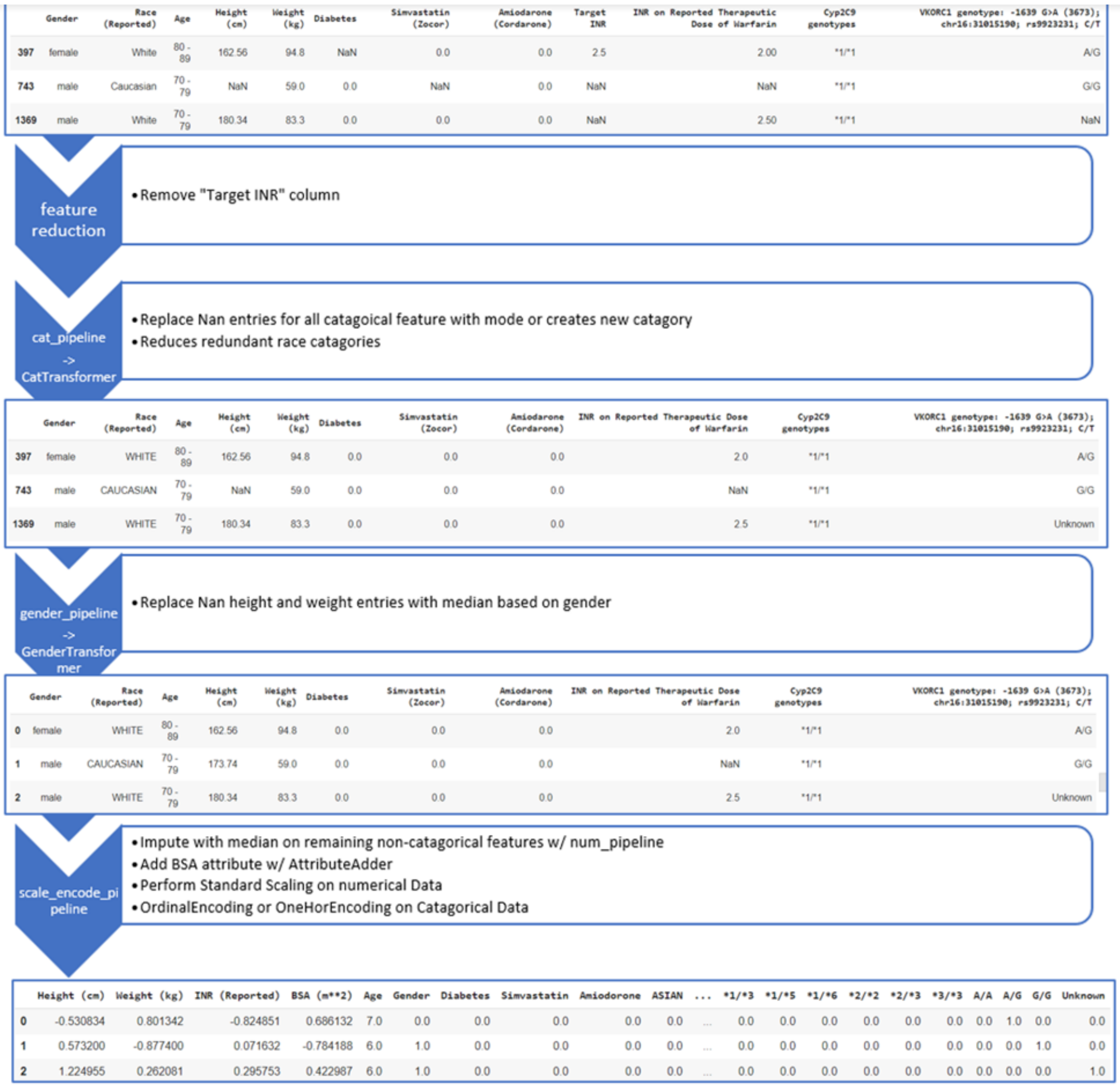


Fig 3: Correlation matrix between Height, Weight, BSA, and BMI attributes on training set. Figure drawn by seaborn


feature reduction
cat_pipeline -> CatTransformer
gender_pipeline -> GenderTransformer
scale_encode_pipeline

	Gender	Race (Reported)	Age	Height (cm)	Weight (kg)	Diabetes	Simvastatin (Zocor)	Amiodarone (Cordarone)	Target INR	INR on Reported Therapeutic Dose of Warfarin	Cyp2C9 genotypes	VKORC1 genotype: -1639 G>A (3673); chr16:31015190; rs9923231; C/T
397	female	White	80 - 89	162.56	94.8	NaN	0.0	0.0	2.5	2.00	*1/*1	A/G
743	male	Caucasian	70 - 79	NaN	59.0	0.0	NaN	0.0	NaN	NaN	*1/*1	G/G
1369	male	White	70 - 79	180.34	83.3	0.0	0.0	0.0	NaN	2.50	*1/*1	NaN

- Remove "Target INR" column

- Replace Nan entries for all catagoical feature with mode or creates new catagory
- Reduces redundant race catagories

	Gender	Race (Reported)	Age	Height (cm)	Weight (kg)	Diabetes	Simvastatin (Zocor)	Amiodarone (Cordarone)	INR on Reported Therapeutic Dose of Warfarin	Cyp2C9 genotypes	VKORC1 genotype: -1639 G>A (3673); chr16:31015190; rs9923231; C/T
397	female	WHITE	80 - 89	162.56	94.8	0.0	0.0	0.0	2.0	*1/*1	A/G
743	male	CAUCASIAN	70 - 79	NaN	59.0	0.0	0.0	0.0	NaN	*1/*1	G/G
1369	male	WHITE	70 - 79	180.34	83.3	0.0	0.0	0.0	2.5	*1/*1	Unknown

- Replace Nan height and weight entries with median based on gender

	Gender	Race (Reported)	Age	Height (cm)	Weight (kg)	Diabetes	Simvastatin (Zocor)	Amiodarone (Cordarone)	INR on Reported Therapeutic Dose of Warfarin	Cyp2C9 genotypes	VKORC1 genotype: -1639 G>A (3673); chr16:31015190; rs9923231; C/T
0	female	WHITE	80 - 89	162.56	94.8	0.0	0.0	0.0	2.0	*1/*1	A/G
1	male	CAUCASIAN	70 - 79	173.74	59.0	0.0	0.0	0.0	NaN	*1/*1	G/G
2	male	WHITE	70 - 79	180.34	83.3	0.0	0.0	0.0	2.5	*1/*1	Unknown

- Impute with median on remaining non-catagorical features w/ num_pipeline
- Add BSA attribute w/ AttributeAdder
- Perform Standard Scaling on numerical Data
- OrdinalEncoding or OneHorEncoding on Catagorical Data

	Height (cm)	Weight (kg)	INR (Reported)	BSA (m**2)	Age	Gender	Diabetes	Simvastatin	Amiodorone	ASIAN	...	*1/*3	*1/*5	*1/*6	*2/*2	*2/*3	*3/*3	A/A	A/G	G/G	Unknown
0	-0.530834	0.801342	-0.824851	0.686132	7.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
1	0.573200	-0.877400	0.071632	-0.784188	6.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
2	1.224955	0.262081	0.295753	0.422987	6.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Fig 4: Flowchart showing preprocessing pipeline as performed by full_preprocessing_function()