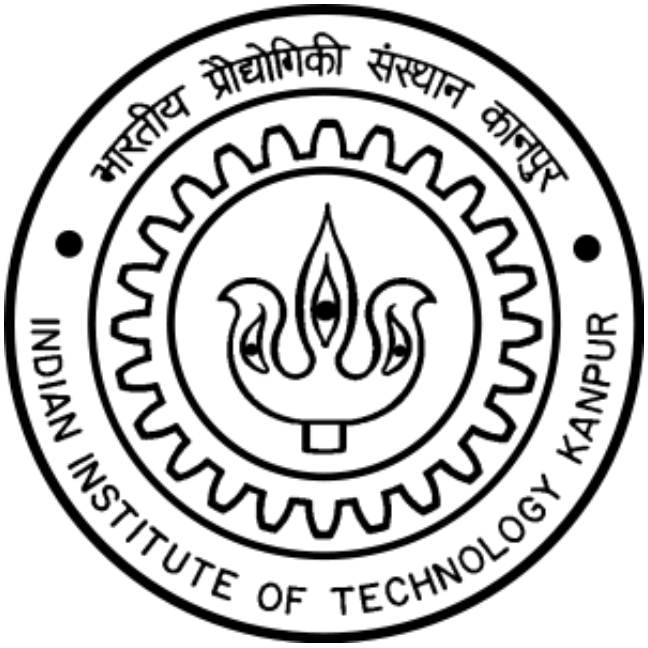
Indian Institute of Technology Kanpur

# Assignment-1 report

*Course Instructor: Prof. Hamim Zafar*

April 21, 2023



## Team Members :

**Ayush Pande**
Roll No.: 20111404
Email id: ayushp20@iitk.ac.in

**Abhishek Kumar Saini**
Roll No.: 20111401
Email id: abhik20@iitk.ac.in

**Yatin**
Roll No.:170824
Email id: yatinaz@iitk.ac.in

## Contribution of Team members :

We had 10 individual drugs to work on, from which we distribute drugs among overselves and work on those drugs individually .
After we tried all approaches for our drugs we also tried our approaches for other members drugs ,if we are able to improve *test score* or not .

# Abhishek's work :
# RIF Dataset :

There are 222 features and 3393 MTB isolates present in our Dataset and we have a another csv file which contain output of each drug *(resistant (1), susceptible (0) and not available (-1)* from which we seperate our $RIF$ column to work with it independently. Also, we dropped all rows where $-1$ occur in output of $RIF$ and then train our model.

# Machine Learning Prediction :

## StratifiedKFold :

```
|: from sklearn.model_selection import StratifiedKFold
   folds = StratifiedKFold(n_splits=5)
```

To assemble a balanced dataset we use *StratifiedKFold* with 5 folds. It split our training data into 5 folds and make sure each fold is representative of original data.Five-fold StratifiedKFold within the training set is conducted to optimise the parameters of each classifier in terms of predictive accuracy.

## Approaches used and performance of each approach :

## Support Vector Classification :

SVM tries to find a separating hyperplane betweentwo class of labelled data points. The hyperplane's location is determined by maximising the distance between it and the closest training data points from each class, which are termed the support vectors. The hyperparameter $C$ needs to be tuned here , also we use *Gausian* kernel with degree 4.

```
model = SVC(C=7, kernel='rbf', degree=4, gamma='auto',probability=True)
```

For these Parameters SVC gives 98.535 *score* on test data.

### GradientBoostingClassifier :

Gradient Boosting Classifier gives prediction model in the form of a set of weak models, which are usually in the form of decision trees. Successive trees are generated during the learning process. This algorithm builds the first model to predict the value and calculate the loss, which is the difference between the first model's result and the actual value. A second model is then built to predict the loss after the first step. This process continues till satisfactory result is achieved.The main idea behind gradient boosting is to iteratively find new trees that minimize the loss function. The loss function is a measure of how large the errors your model makes.
Here we have several parameters which needs tobe tuned *learning rate, nestimators, maxfeatures, maxdepth*

```
model = GradientBoostingClassifier(n_estimators=75, learning_rate=1.38, max_features=2, max_depth=2, random_state=8
```

For these parametres GBC performs suitably well gives 98.673 *score* on test data.

### XGBClassifier :

XGBoost is an ensemble method. The trees are constructed iteratively until a stopping criterion is met.XGBoost is a more regularized form of Gradient Boosting. XGBoost uses advanced regularization, which improves model generalization capabilities.

```
model = xgb.XGBClassifier(random_state=1,learning_rate=0.33)
```

For these parameters XGBoost gives 98.301 *score*.

### VotingClassifier For Different Models:

On using Stratified K-fold some models gives less accuracy on a fold while other model gives good accuracy on the same fold so we tried Voting Classifier using 5 models named as *SVC, GradientBoostingClassifier, AdaBoostClassifier, XGBClassifier, RandomForestClassifier*

```
VotingClassifier(estimators=[('SVM', model1), ('GBC', model2) , ('ABC', model3), ('XGB', model4), ('RFC', model5)],
```

But it doesn't give any significance improvement over *GBC*

### Artificial Neural Network :

Inthis model, I split the training dataset into 2 parts. 90 *percent* is used for training purpose and 10 percent is used for cross-validation and test set.

***Architecture for Neural Network*** :

```
model = tf.keras.models.Sequential([
tf.keras.layers.InputLayer(input_shape=219),
tf.keras.layers.Dense(128,activation='relu'),
tf.keras.layers.Dense(16,activation='relu'),
tf.keras.layers.Dense(4,activation='relu'),
tf.keras.layers.Dense(1,activation='sigmoid'),
])
```

After running it several times on different number of epochs, it gives 98.834 *score* on test data.

## Comparision :

| Model | Score |
|---|---|
| $SupportVectorClassifier$ | 98.535 |
| $GradientBoostingClassifier$ | 98.878 |
| $XGBClassifier$ | 98.301 |
| $UsingVotingClassifier(on5models)$ | 97.322 |
| $ArtificialNeuralNetwork$ | 98.834 |

## STR Dataset :

There are 222 features and 3393 MTB isolates present in our Dataset and we have a another csv file which contain output of each drug *(resistant (1), susceptible (0) and not available (-1)* from which we seperate our $STR$ column to work with it independently. Also, we dropped all rows where $-1$ occur in output of $STR$ and then train our model.

## Machine Learning Prediction :

### StratifiedKFold :

We used StratifiedKFold with 5 folds within the training set to optimise the parameters of each classifier in terms of predictive accuracy.

### Support Vector Classification :

First we use Support vector classifier and use **GridSearch** for hyperparametre estimation

```
model = SVC(C=37, kernel='rbf', degree=4, gamma='auto',probability=True)
```

For these Parameters SVC gives 95.155 *score* on test data.

### GradientBoostingClassifier : :

After SVC we use GradientBoosting classifier which give significance improvement over SVC, also we tried to use **iterativeImputer** here to fill $-1$ entries using **KNeighborsRegressor** estimater of iterative imputer , but the problem with this imputer is that it is very uncertain and gives different result on same model(*with same parameters*). so we didn't used it in while predicting test score.
on tuning GBC parametres as,

```
GradientBoostingClassifier(n_estimators=250, learning_rate=0.93, max_features=4, max_depth=4, random_state=1,subsar
```

For these Parameters GBC gives 93.726 *score* on test data.

### AdaBoostClassifier : :

In AdaBoostClassfier Weak models are added sequentially, trained using the weighted training data.The process continues until a pre-set number of weak learners have been created (parameter) or no further improvement can be made on the training dataset. AdaBoost doesn't perfrom significantly well here in comparision to other models.

### XGBClassifier :

We then used another ensemnle learning approach which perform better than other models for this drug. On tuning hyperparameters(*lamda and gamma*) using **GridSearch** we set our parameters as,

```
xgb.XGBClassifier(random_state=100,learning_rate=0.99,gamma=0.16)
```

For these parameters XGBoost gives 96.040 *score*.

## Comparision :

| Model | Score |
|---|---|
| $SupportVectorClassifier$ | 95.155 |
| $GradientBoostingClassifier$ | 93.726 |
| $XGBClassifier$ | 96.040 |
| $AdaBoostClassifier$ | 92.373 |

# MOXY Dataset :

There are 222 features and 3393 MTB isolates present in our Dataset and we have a another csv file which contain output of each drug *(resistant (1), susceptible (0) and not available (-1)* from which we seperate our $STR$ column to work with it independently. Also, we dropped all rows where $-1$ occur in output of $STR$ and then train our model.

## Class imbalance :

After dropping all rows where $-1$ occur as output. we have

| Label | Total |
|---|---|
| 0 | 267 |
| 1 | 1070 |

Clearly this is a class imbalance, we try to use a Library called **SMOTETomek** for **UpSampling**. After UpSampling we have

| Label | Total |
|---|---|
| 0 | 1058 |
| 1 | 1058 |

# Machine Learning Prediction :

## StratifiedKFold :

We used StratifiedKFold with 5 folds within the training set to optimise the parameters of each classifier in terms of predictive accuracy.

### Support Vector Classification :

First we use Support vector classifier and use **GridSearch** for hyperparametre estimation , we tried $C$ value from $1 to 100$ , but the problem here is because of more values of *label 1* it overfits for label 1(*even after upsampling using* ***SMOTETomek***).

### GradientBoostingClassifier : :

After SVC we use GradientBoosting classifier which doesn't give much improvement over SVC and having same problem as SVC because of class imbalance problem, it is more biased towards labels 1 .

### AdaBoostClassifier : :

In AdaBoostClassfier Weak models are added sequentially, trained using the weighted training data and this approach give significant improvement over previos models(*with upsampling*)

```
model = AdaBoostClassifier(n_estimators=75, learning_rate=0.99, random_state=0)
```

### XGBClassifier :

We then used another ensemnle learning approach which perform better than other models for this drug. On tuning hyperparameters(*lamda and gamma*) using **GridSearch** we set our parameters as,

```
model = xgb.XGBClassifier(random_state=50,learning_rate=0.07)
```

**Without UpSampling :** For these parameters XGBoost gives 97.071 *score*.
**With UpSampling using SMOTETomek :** For these parameters it gives 98.061 *score*.

## Comparision :

| Model | Score |
|---|---|
| $SupportVectorClassifier$ | 91.562 |
| $GradientBoostingClassifier$ | 90.505 |
| $AdaBoostClassifier$ | 92.151 |
| $XGBClassifier(withoutupsampling)$ | 97.071 |
| $XGBClassifier(withupsampling)$ | 98.061 |

## Ayush Pande's Work-:

We'll begin by covering one by the one the theoretical aspects of the models we've used and then go on to describe our individual contributions.

### Data Preprocessing:

Removed following columns of the traindata containing -1.

'SNP_CN_2714366_C967A_V323L_eis'

'SNP_I_2713795_C329T_inter_Rv2415c_eis'

'SNP_I_2713872_C252A_inter_Rv2415c_eis'

Tried to find the correlation between the feature columns and the target variable for Dimensionality Reduction.I used the correlation matrix to find the insights in the data.But it didn't workout.

### Training Process and Steps for solving the challenges:

1. I have used google colab for training and testing all the models.I would request to use Google Colab,for running and evaluating the notebooks.
2. For XGBCLASSIFIER I have used GPU for speeding up the training process.
3. Before,starting the training and checking the models,I tried to figure out the magnitude of class imbalance in the target variable.
4. Next, I tried to find the correlation between the feature vectors using correlation matrix.But,it didn't work out in none of the drugs I tried.
5. Then,I have applied different ML models and most of the time was spent in there hyperparameter tuning.Mostly,the XGBCLASSIFIER took lot of time even after using GPU.For,hyperparameter tuning I mostly used GridSearchCV because it returns the most optimal combination of all the hyperparameter among the set of parameters specified.
6. Also,there was a big difference in score of models on the training set and the test set.So, I have to make multiple submissions to figure out the tuning of hyperparameters which I am doing is going in the right direction or not.

## Description about the ML models used:

1. Logistic Regression:

   Logistic Regression is used when the dependent variable(target) is categorical.And,here we have to do binary classification for each drug and predict the value 0 or 1.So,I used this model. Important Hyperparameters of this model:

   (a) **"C" float, default=1.0**

   Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.

   (b) `class_weight` dict or 'balanced', default=None

   Weights associated with classes in the form `class_label`: weight. If not given, all classes are supposed to have weight one.

   The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as `n_samples / (n_classes * np.bincount(y))`

   (c) **solver**'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga', default='lbfgs'

   Algorithm to use in the optimization problem.

   (d) `max_iter` int, default=100

   Maximum number of iterations taken for the solvers to converge.

2. Support Vector Machine(SVM): The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence. Important Hyperparameters of this model:

   (a) **"C" float, default=1.0**

   Inverse of regularization strength; must be a positive float.Smaller values specify stronger regularization.

   (b) **kernel** 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed', default='rbf'

   Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable. If none is given, 'rbf' will be used. If a

callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape `(n_samples, n_samples)`.

(c) **probability** bool, default=False

This was set True in all cases because we need to predict the probability between 0 and 1 for the data sample.

(d) `class_weight` dict or 'balanced', default=None

Set the parameter C of class i to `class_weight[i]*C` for SVC. If not given, all classes are supposed to have weight one. The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as `n_samples / (n_classes * np.bincount(y))`

3. RandomForestClassifier

It is an ensemble tree-based learning algorithm. The Random Forest Classifier is a set of decision trees from randomly selected subset of training set. It aggregates the votes from different decision trees to decide the final class of the test object.

Features and Advantages of Random Forest :

It is one of the most accurate learning algorithms available. For many data sets, it produces a highly accurate classifier. It runs efficiently on large databases. It can handle thousands of input variables without variable deletion. It gives estimates of what variables that are important in the classification.

Important Hyperparameters of this model:

(a) `n_estimators` int, default=100

The number of trees in the forest.

(b) `max_depth` int, default=None

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than `min_samples_split` samples.

4. XGBClassifier

With a regular machine learning model, like a decision tree, we'd simply train a single model on our dataset and use that for prediction. We might play around with the parameters for a bit or augment the data, but in the end we are still using a single model. Even if we build

an ensemble, all of the models are trained and applied to our data separately. Boosting, on the other hand, takes a more iterative approach. It's still technically an ensemble technique in that many models are combined together to perform the final one, but takes a more clever approach. Rather than training all of the models in isolation of one another, boosting trains models in succession, with each new model being trained to correct the errors made by the previous ones. Models are added sequentially until no further improvements can be made. The advantage of this iterative approach is that the new models being added are focused on correcting the mistakes which were caused by other models. In a standard ensemble method where models are trained in isolation, all of the models might simply end up making the same mistakes!

Important Hyperparameters of this model:

(a) `max_depth` [default=6]

Maximum depth of a tree. Increasing this value will make the model more complex

(b) `scale_pos_weight`[default=1]

Control the balance of positive and negative weights, useful for unbalanced classes. A typical value to consider: sum(negative instances) / sum(positive instances)

(c) **objective** [default=reg:squarederror]

I used binary:logistic because logistic regression for binary classification, output probability.

(d) `learning_rate` [default=0.1] This decides the rate of descent of the iterative algorithm to converge towards the optima. I contributed in the following drugs:

## Drug PZA:

Since there is class imbalance in the data for this drug where we have 2246 labels as 1 and 695 labels as 0.

Approach 1:
I tried ensemble method for this.I break the 80% train dataset of label '1' into 3 parts and remaining 20% as test set of label '1' as follows:
Part 1: 626 samples
Part 2: 626 samples
Part 3: 535 samples
Next,I made the dataset for the 3 SVM classifiers containing approximately equal amounts of data of label '1' and label '0 for ensemble training.
So, by this each classifier will properly fit for each of the class and there will be no skewness in the model for the label '1' class.
I joined the results of these models to give the final predictions using another SVM classifier.But,this method gave a score of about 0.81.

Approach 2:
In,this approach I tried different ML models to find out which one of them performs better in our case.For all the models I kept 90% train data and 10% test data split.
I used GridSearchCV,RandomSearchCV for hyperparameter tuning.
I used following models for this drug:

1. Logistic Regression
   Following is the code with the given hyperparameters which gave the best accuracy for this classifier after hyperparameter tuning

   ```
   LogisticRegression(C=1.7, class_weight='balanced',max_iter=20, solver='saga')
   ```

   Accuracy on Test set of Training Data: **0.9016949152542373**

Confusion Matrix

$$\begin{bmatrix} 65 & 5 \\ 24 & 201 \end{bmatrix}$$

I got a **0.89740** score on submission of the predictions of this model on the Kaggle competition.

2. SVM

   Following is the code with the given hyperparameters which gave the best accuracy for this classifier after hyperparameter tuning

   ```
   SVC(C=0.5,class_weight='',probability=True)
   ```

   Accuracy on Test set of Training Data: **0.9254237288135593**

   Confusion Matrix

   $$\begin{bmatrix} 59 & 11 \\ 11 & 214 \end{bmatrix}$$

   I got a **0.90593** score on submission of the predictions of this model on the Kaggle competition.

3. RandomForestClassifier

   Following is the code with the given hyperparameters which gave the best accuracy for this classifier after hyperparameter tuning

   ```
   RandomForestClassifier(n_estimators=55,random_state=0,max_depth=15)
   ```

   Accuracy on Test set of Training Data: **0.9288135593220339**

   Confusion Matrix

   $$\begin{bmatrix} 60 & 10 \\ 11 & 214 \end{bmatrix}$$

   I got a **0.92479** score on submission of the predictions of this model on the Kaggle competition.

4. XGBClassifier

   Following is the code with the given hyperparameters which gave the best accuracy for this classifier after hyperparameter tuning

   ```
   XGBClassifier(n_estimators=50,max_depth=4,learning_rate=0.1)
   ```

   Accuracy on Test set of Training Data: **0.9389830508474576**

   Confusion Matrix

   $$\begin{bmatrix} 58 & 12 \\ 6 & 219 \end{bmatrix}$$

   I got a **0.90870** score on submission of the predictions of this model on the Kaggle competition.

Approach 3:

In,this approach I took the help of Artificial Neural Net to increase the score attained by RandomForestClassifier of about **0.92479**.

I split the train dataset into 2 parts.90% is used for training purposes and remaining 10% is used for cross-validation and test set.

Following is the architecture of neural network used for this drug:

model = tf.keras.models.Sequential([
tf.keras.layers.InputLayer(input_shape=219),
tf.keras.layers.Dense(128,activation='relu'),
tf.keras.layers.Dense(16,activation='relu'),
tf.keras.layers.Dense(4,activation='relu'),
tf.keras.layers.Dense(1,activation='sigmoid'),
])

The optimizer and loss function used is in the following code:

```
model.compile(loss='binary_crossentropy',optimizer=tf.keras.optimizers.Adam())
```

```
model.fit(X_train,y_train,epochs=3,validation_data=(X_test,y_test))
```

Accuracy on Test set of Training Data: **0.9152542372881356**

Confusion Matrix

$$\begin{bmatrix} 52 & 18 \\ 7 & 218 \end{bmatrix}$$

I run about 3 epochs for this and then submit the predictions on the kaggle platform.The score by this model was the highest among all other models of about **0.93447**.

I also tried setting the class weights in which model is penalised equally for minority class as that of majority class during model training to handle the class imbalance but it gave a low score of about **0.92918**.

See the following code for class weight setting:
weights = class_weight.compute_class_weight('balanced',np.unique(y_train),y_train)
temp = 0:weights[0],1:weights[1]

```
model.fit(X_train,y_train,epochs=3,validation_data=(X_test,y_test),class_weight=temp)
```

## Drug CAP :

I didn't use Ensemble approach as I used it in drug PZA and it didn't give any good results.

I used GridSearchCV,RandomSearchCV for hyperparameter tuning.

Approach 1:
In,this approach I tried different ML models to find out which one of them performs better in our case.For all the models I kept 90% train data and 10% test data split.

I used followiung models for this drug.

1. LogisticRegression
   Following is the code with the given hyperparameters which gave the best accuracy for this classifier after hyperparameter tuning

   ```
   LogisticRegression(C=0.3, class_weight='balanced', dual=False, fit_intercept=True,
   intercept_scaling=1, l1_ratio=None, max_iter=250, multi_class='auto', n_jobs=None,
   penalty='l2', random_state=None, solver='saga', tol=0.0001, verbose=0, warm_start=False)
   ```

   Accuracy on Test set of Training Data: **0.8582089552238806**

   Confusion Matrix

   $$\begin{bmatrix} 52 & 7 \\ 12 & 63 \end{bmatrix}$$

   I got a **0.83143** score on submission of the predictions of this model on the Kaggle competition.

2. SVM

   Following is the code with the given hyperparameters which gave the best accuracy for this classifier after hyperparameter tuning

   ```
   SVC(gamma='scale',C=1.7,kernel='rbf',class_weight='balanced')
   ```

   Accuracy on Test set of Training Data: **0.9067164179104478**

   Confusion Matrix

   $$\begin{bmatrix} 109 & 8 \\ 17 & 134 \end{bmatrix}$$

   I got a **0.81310** score on submission of the predictions of this model on the Kaggle competition.

3. RandomForestClassifier

   Following is the code with the given hyperparameters which gave the best accuracy for this classifier after hyperparameter tuning

   ```
   RandomForestClassifier(n_estimators=50,max_depth=14)
   ```

   Accuracy on Test set of Training Data: **0.9029850746268657**

   Confusion Matrix

   $$\begin{bmatrix} 54 & 5 \\ 8 & 67 \end{bmatrix}$$

   I got a **0.776** score on submission of the predictions of this model on the Kaggle competition.

4. XGBClassifier

   Following is the code with the given hyperparameters which gave the best accuracy for this classifier after hyperparameter tuning

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bynode=1,
colsample_bytree=0.5, gamma=0.0, gpu_id=0, importance_type='gain', interaction_constraints='',
learning_rate=0.05, max_delta_step=0, max_depth=12, min_child_weight=1, missing=None,
n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0, reg_alpha=0, reg_lambda=1,
scale_pos_weight=1, subsample=1, validate_parameters=1)
```

   Accuracy on Test set of Training Data: **0.8731343283582089**

   Confusion Matrix

$$\begin{bmatrix} 48 & 11 \\ 6 & 69 \end{bmatrix}$$

   I got a **0.77953** score on submission of the predictions of this model on the Kaggle competition.

Approach 2:
In,this approach I took the help of Artificial Neural Net to increase the score attained by LogisticRegression of about **0.83143**.

I split the train dataset into 2 parts.90% is used for training purposes and remaining 10% is used for cross-validation and test set. Following is the architecture of neural network used for this drug:
model = tf.keras.models.Sequential([
tf.keras.layers.InputLayer(input_shape=219),
tf.keras.layers.Dense(32,activation='relu'),
tf.keras.layers.Dense(16,activation='relu'),
tf.keras.layers.Dense(8,activation='relu'),
tf.keras.layers.Dense(4,activation='relu'),

tf.keras.layers.Dense(1,activation='sigmoid'),
])

The optimizer and loss function used is in the following code:

```
model.compile(loss='binary_crossentropy',optimizer=tf.keras.optimizers.Adam())
```

Accuracy on Test set of Training Data: **0.8582089552238806**

Confusion Matrix

$$\begin{bmatrix} 51 & 8 \\ 11 & 64 \end{bmatrix}$$

The score by this model was the highest among all other models of about **0.86465**.
I don't remember the number of epochs exactly but it was less then 8 and validation loss was less than 0.3.

I also tried setting the class weights in which model is penalised equally for minority class as that of majority class during model training to handle the class imbalance but it gave low score of about **0.81477**.

See the following code for class weight setting:

```
weights = class_weight.compute_class_weight('balanced',np.unique(y_train),y_train)
temp = 0:weights[0],1:weights[1]

model.fit(X_train,y_train,epochs=3,validation_data=(X_test,y_test),class_weight=temp)
```

## Drug KAN:

I didn't use Ensemble approach as I used it in drug PZA and it didn't give any good results.

I used GridSearchCV,RandomSearchCV for hyperparameter tuning.

Approach 1:
In,this approach I tried different ML models to find out which one of them performs better in our case.For all the models I kept 90% train data and 10% test data split.

1. LogisticRegression
   Following is the code with the given hyperparameters which gave the best accuracy for this classifier after hyperparameter tuning

   ```
   LogisticRegression(C=0.1, class_weight='balanced', dual=False, fit_intercept=True,
   intercept_scaling=1, l1_ratio=None, max_iter=60, multi_class='auto', n_jobs=None,
   penalty='l2', random_state=None, solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
   ```

   Accuracy on Test set of Training Data: **0.8604651162790697**

   Confusion Matrix

   $$\begin{bmatrix} 16 & 11 \\ 7 & 95 \end{bmatrix}$$

   I got a **0.90543** score on submission of the predictions of this model on the Kaggle competition.

2. SVM
   Following is the code with the given hyperparameters which gave the best accuracy for this classifier after hyperparameter tuning

```
SVC(gamma='scale',C=5,kernel='rbf',class_weight='',probability=True,decision_function_shape='o
```

Accuracy on Test set of Training Data: **0.8837209302325582**

Confusion Matrix

$$\begin{bmatrix} 13 & 14 \\ 1 & 101 \end{bmatrix}$$

I got a **0.90285** score on submission of the predictions of this model on the Kaggle competition.

3. RandomForestClassifier

   Following is the code with the given hyperparameters which gave the best accuracy for this classifier after hyperparameter tuning

   RandomForestClassifier(n_estimators=130,max_depth=4)

   Accuracy on Test set of Training Data: **0.8217054263565892**

   Confusion Matrix

$$\begin{bmatrix} 5 & 22 \\ 1 & 101 \end{bmatrix}$$

   I got a **0.898 to 0.92191** score on multiple submissions of the predictions of this model on the Kaggle competition.

4. XGBClassifier

   Following is the code with the given hyperparameters which gave the best accuracy for this classifier after hyperparameter tuning

```
XGBClassifier(learning_rate=0.1, max_depth=6, n_estimators=60)
```

Accuracy on Test set of Training Data: **0.875968992248062**

Confusion Matrix

$$\begin{bmatrix} 13 & 14 \\ 2 & 100 \end{bmatrix}$$

I got a **0.895** score on submission of the predictions of this model on the Kaggle competition.

Approach 2:
In,this approach I took the help of Artificial Neural Net to increase the score attained by RandomForestClassifier of about **0.92191**.

I split the train dataset into 2 parts.90% is used for training purposes and remaining 10% is used for cross-validation and test set.
Following is the architecture of neural network used for this drug:

model = tf.keras.models.Sequential([
tf.keras.layers.InputLayer(input_shape=219),
tf.keras.layers.Dense(96,activation='relu'),
tf.keras.layers.Dense(4,activation='relu'),
tf.keras.layers.Dense(1,activation='sigmoid'),
])

The optimizer and loss function used is in the following code:

model.compile(loss='binary_crossentropy',optimizer=tf.keras.optimizers.Adam())

Accuracy on Test set of Training Data: **0.8837209302325582**

Confusion Matrix

$$\begin{bmatrix} 13 & 14 \\ 1 & 101 \end{bmatrix}$$

The score by this model was not highest among all other models of about **0.91216**.
I don't remember the number of epochs exactly but it was less then 8 and validation loss was less than 0.4.

I also tried setting the class weights in which model is penalised equally for minority class as that of majority class during model training to handle the class imbalance but it gave low score of about **0.88680**.

See the following code for class weight setting:

```
weights = class_weight.compute_class_weight('balanced',np.unique(y_train),y_train)
```

temp = 0:weights[0],1:weights[1]

```
model.fit(X_train,y_train,epochs=3,validation_data=(X_test,y_test),class_weight=temp)
```

## Drug EMB:

I didn't use Ensemble approach as I used it in drug PZA and it didn't give any good results.

I used GridSearchCV,RandomSearchCV for hyperparameter tuning.

Approach 1:
In,this approach I tried different ML models to find out which one of them performs better in our case.For all the models I kept 90% train data and 10% test data split.

1. LogisticRegression
   Following is the code with the given hyperparameters which gave the best accuracy for this classifier after hyperparameter tuning

   ```
   LogisticRegression(C=1, class_weight='balanced', dual=False, fit_intercept=True,
   intercept_scaling=1, l1_ratio=None, max_iter=20, multi_class='auto', n_jobs=None,
   penalty='l2', random_state=None, solver='sag', tol=0.0001, verbose=0, warm_start=False)
   ```

   Accuracy on Test set of Training Data : **0.9216867469879518**

   Confusion Matrix

   $$\begin{bmatrix} 92 & 5 \\ 21 & 214 \end{bmatrix}$$

   I got a **0.93380** score on submission of the predictions of this model on the Kaggle competition.

2. SVM
   Following is the code with the given hyperparameters which gave the best accuracy for this classifier after hyperparameter tuning

```
SVC(C=2, break_ties=False, cache_size=200, class_weight='balanced', coef0=0.0, decision_functio
degree=1, gamma='scale', kernel='linear', max_iter=-1, probability=True, random_state=None,
shrinking=True, tol=0.001, verbose=False)
```

Accuracy on Test set of Training Data: **0.9186746987951807**

Confusion Matrix

$$\begin{bmatrix} 89 & 8 \\ 19 & 216 \end{bmatrix}$$

I got a **0.93015** score on submission of the predictions of this model on the Kaggle competition.

3. RandomForestClassifier

   Following is the code with the given hyperparameters which gave the best accuracy for this classifier after hyperparameter tuning

```
RandomForestClassifier(max_depth=17, max_features='auto', max_leaf_nodes=None, max_samples=None
min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=4,
min_weight_fraction_leaf=0.0, n_estimators=50, n_jobs=None, oob_score=False, random_state=0,
verbose=0, warm_start=False)
```

   Accuracy on Test set of Training Data: **0.9216867469879518**

   Confusion Matrix

$$\begin{bmatrix} 90 & 7 \\ 19 & 216 \end{bmatrix}$$

   I got a **0.93257** score on submission of the predictions of this model on the Kaggle competi-

tion.

4. XGBClassifier

   Following is the code with the given hyperparameters which gave the best accuracy for this classifier after hyperparameter tuning

   ```
   XGBClassifier( learning_rate=0.1, max_delta_step=0, max_depth=6, n_estimators=130)
   ```

   Accuracy on Test set of Training Data: **0.9216867469879518**

   Confusion Matrix

   $$\begin{bmatrix} 89 & 8 \\ 18 & 217] \end{bmatrix}$$

   I got a **0.94097** score on submission of the predictions of this model on the Kaggle competition.

Approach 2: In,this approach I took the help of Artificial Neural Net to increase the score attained by RandomForestClassifier of about **0.94097**.

I split the train dataset into 2 parts.90% is used for training purposes and remaining 10% is used for cross-validation and test set.
Following is the architecture of neural network used for this drug:

```
model = tf.keras.models.Sequential([
tf.keras.layers.InputLayer(input_shape=219),
tf.keras.layers.Dense(128,activation='relu'),
tf.keras.layers.Dense(16,activation='relu'),
tf.keras.layers.Dense(4,activation='relu'),
tf.keras.layers.Dense(1,activation='sigmoid'),
])
```

The optimizer and loss function used is in the following code:

```
model.compile(loss='binary_crossentropy',optimizer=tf.keras.optimizers.Adam())
```

Accuracy on Test set of Training Data: **0.9126506024096386**

Confusion Matrix

$$\begin{bmatrix} 84 & 13 \\ 16 & 219 \end{bmatrix}$$

I run about 3 epochs for this and then submit the predictions on the kaggle platform.
The score by this model was the highest among all other models of about **0.94757**.
I also tried setting the class weights in which model is penalised equally for minority class as that of majority class during model training to handle the class imbalance but it gave low score of about **0.94681**.

See the following code for class weight setting:
weights = class_weight.compute_class_weight('balanced',np.unique(y_train),y$_train$)
$temp = 0 : weights[0], 1 : weights[1]$

$model.fit(X\_train, y\_train, epochs = 3, validation\_data = (X\_test, y\_test), class\_weight = temp)$

## Yatin's work :

## OFLX Dataset :

As before there are 3393 isolates and 222 features in our dataset. Information on drug resistance is available for only 690 isolates. There is also class imbalance since the proportion of drug resistance and susceptibility labels is different. We separate out the -1 labels initially from the dataset, since there's no real way to predict for those labels.

### Class imbalance :

After dropping all rows where $-1$ occur as output. we have

| Label | Total |
|---|---|
| 0 | 87 |
| 1 | 603 |

### Support Vector Classification :

First I tried Support vector classifier and used **GridSearchCV** for hyperparameter tuning , I tried **C** values from 0.01 to 15 and **gamma** values from 0.01 to 10. As already discussed before, SVC did not do too well because of the prevalent class imbalance in the problem. The optimal C was found to be 10 and gamma was found to be 0.1. I report below the the classification metrics as well as the auc_roc score which on the training set. The split between the training and test samples was 0.1. That is, 90% of the training data was used for training and 10% for testing.

### Random Forest Classifier:

Knowing that Random Forest Classifiers work well with imbalanced classes, I decided to use it for the given drug. Hyperparameter tuning was needed and I achieved that with random search CV. The number of estimators, maximum depth, minimum samples at the leaf node, minimum samples needed for split of the tree were varied within a given range and their optimal values were determined.

Classification metrics for SVM with OFLX

```
[[ 6  6]
 [ 0 57]]
              precision    recall  f1-score   support

           0       1.00      0.50      0.67        12
           1       0.90      1.00      0.95        57

    accuracy                           0.91        69
   macro avg       0.95      0.75      0.81        69
weighted avg       0.92      0.91      0.90        69

0.75
```

Eventually, these were the optimal values determined for each of the parameters.

| Parameter | Value |
|---|---|
| n_estimators | 567 |
| max_depth | 208 |
| min_samples_leaf | 1 |
| min_samples_split | 2 |
| bootstrap | False |

The auc_roc_score for this model on training set of OFLX is **0.83334** The random forest classifier

Classification metrics for RCV Random Forest with OFLX

```
[[ 8  4]
 [ 0 57]]
              precision    recall  f1-score   support

           0       1.00      0.67      0.80        12
           1       0.93      1.00      0.97        57

    accuracy                           0.94        69
   macro avg       0.97      0.83      0.88        69
weighted avg       0.95      0.94      0.94        69

0.8333333333333334
```

worked well sometimes, but the performance wasn't very reliable and the peak auc_roc_score achieved on the test data set was only around 0.879 with different combinations of hyperparameters used.

## XGBoost Classifier :

As explained before as well XGBoost is another ensemble learning approach that makes use of multiple weak decision trees parallelly constructed and optimised to achieve classification. This model actually achieved peak performance for this drug both on the test set and the training set. Hyperparameters were tuned using grid search CV and 3 fold cross validation was used for optimising the auc_roc_score.

These were the optimal parameters for XGB classifier

| Parameter | Value |
|---|---|
| learning_rate | 0.3 |
| gamma | 0 |
| min_child_weight | 1 |
| max_delta_step | 1 |
| max_depth | 6 |

Overall, the best performance on the test data set was shown by XGBoost, the performance varied from 0.881 to 0.898. This is because I'm sampling the dataset each time. So, each time I run the grid search CV on my classifier, it produces a different set of optimal parameters. A change in each of these parameters influences the output accuracy slightly.

The **auc_roc_score** for this model on OFLX is **0.83334**

Classification metrics for XGBoost with OFLX

```
[[ 8  4]
 [ 0 57]]
              precision    recall  f1-score   support

           0       1.00      0.67      0.80        12
           1       0.93      1.00      0.97        57

    accuracy                           0.94        69
   macro avg       0.97      0.83      0.88        69
weighted avg       0.95      0.94      0.94        69

0.8333333333333334
```

## Comparison :

For the comparison across models I'm only mentioning the best achieved score with each model

| Model | Score |
|---|---|
| Support Vector Classifier | 0.86381 |
| XGBClassifier | 0.89812 |
| Random Forest | 0.87994 |

## AMK Dataset :

As before there are 3393 isolates and 222 features in our dataset. Information on drug resistance is available for only 1360 isolates. There is also class imbalance since the proportion of drug resistance and susceptibility labels is different. We separate out the -1 labels initially from the dataset, since there's no real way to predict for those labels.

**Class imbalance :** After dropping all rows where -1 occur as output. we have

| Label | Total |
|---|---|
| 0 | 223 |
| 1 | 1127 |

### Support Vector Classification :

First I tried Support vector classifier and used **GridSearchCV** for hyperparameter tuning , I tried **C** values from 0.01 to 10 and **gamma** values from 0.01 to 2. The optimal C was found to be 10 and gamma was found to be 0.2. The radial basis function was used with degree 3. I report below the the classification metrics as well as the auc_roc score which is 0.9286. The split between the training and test samples was 0.1. That is, 90% of the training data was used for training and 10% for testing. SVC performance overfitted on training data with upsampling. Hence, maximum test accuracy was achieved without upsampling only.

Classification metrics for SVM with AMK

```
[[ 18   3]
 [  0 115]]
              precision    recall  f1-score   support

           0       1.00      0.86      0.92        21
           1       0.97      1.00      0.99       115

    accuracy                           0.98       136
   macro avg       0.99      0.93      0.96       136
weighted avg       0.98      0.98      0.98       136

0.9285714285714286
```

## Random Forest Classifier:

Knowing that Random Forest Classifiers work well with imbalanced classes, I decided to use it for the given drug. Hyperparameter tuning was needed and I achieved that with random search CV. The number of estimators, maximum depth, minimum samples at the leaf node, minimum samples needed for split of the tree were varied within a given range and their optimal values were determined. Eventually, these were the optimal values determined for each of the parameters.

| Parameter | Value |
|---|---|
| n_estimators | 1008 |
| max_depth | 23 |
| min_samples_leaf | 1 |
| min_samples_split | 2 |
| bootstrap | False |

The auc_roc_score for this model on AMK is **.9437** This model actually achieved peak performance for this drug both on the test set and the training set. The peak auc_roc_score achieved on the test data set was around 0.9643.

## XGBoost Classifier :

An ensemble approach decision tree approach. It slightly overfitted the training set, hence its

Classification metrics for RCV Random Forest with AMK

```
[[ 19   2]
 [  2 113]]
              precision    recall  f1-score   support

           0       0.90      0.90      0.90        21
           1       0.98      0.98      0.98       115

    accuracy                           0.97       136
   macro avg       0.94      0.94      0.94       136
weighted avg       0.97      0.97      0.97       136

0.9436853002070394
```

performance on the test set wasn't that good. Hyperparameters were tuned using grid search CV and 3 fold cross validation was used for optimising the auc_roc_score.

| Parameter | Value |
|---|---|
| learning_rate | 0.27 |
| gamma | 0 |
| min_child_weight | 1 |
| max_delta_step | 1 |
| max_depth | 6 |

The **auc_roc_score** for this model on AMK test set was **0.8649**

Classification metrics for XGBoost with AMK

```
              precision    recall  f1-score   support

           0       0.90      0.90      0.90        21
           1       0.98      0.98      0.98       115

    accuracy                           0.97       136
   macro avg       0.94      0.94      0.94       136
weighted avg       0.97      0.97      0.97       136

0.9436853002070394
```

Overall, the best performance on the test data set was shown by Random Forest, the performance varied from 0.961 to 0.964. This is because I'm sampling the dataset each time. So, each time I run the Random search CV on my classifier, it produces a different set of optimal parameters. A change in each of these parameters influence the output accuracy slightly.

## Comparison :

For the comparison across models I'm only mentioning the best achieved score with each model

| Model | Score |
|---|---|
| Support Vector Classifier | 0.9356 |
| XGBClassifier | 0.8679 |
| Random Forest | 0.96428 |

## INH Dataset :

As before there are 3393 isolates and 222 features in our dataset. Information on drug resistance is available for a remarkable 3356 isolates. There is also no class imbalance since the proportion of drug resistance and susceptibility labels was approximately equal. We separate out the -1 labels initially from the dataset, since there's no real way to predict for those labels.

### Support Vector Classification :

First I tried Support vector classifier and used **GridSearchCV** for hyperparameter tuning , I tried **C** values from 0.01 to 20 and **gamma** values from 0.005 to 0.1. The optimal C was found to be 3 and gamma was found to be 0.03. The radial basis function was used with degree 3. I report below the the classification metrics as well as the auc_roc score which is 0.9706. The split between the training and test samples was 0.2. That is, 80% of the training data was used for training and 20% for testing. SVC performance overfitted on training data with upsampling. Hence, maximum test accuracy was achieved without upsampling only.
This model achieved peak accuracy on the training set but not on the test data set.

### Random Forest Classifier:

Random Forest Classifier is another model I used for my drug. Hyperparameter tuning was needed

Classification metrics for SVM with INH

```
[[295  13]
 [  6 358]]
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       308
           1       0.96      0.98      0.97       364

    accuracy                           0.97       672
   macro avg       0.97      0.97      0.97       672
weighted avg       0.97      0.97      0.97       672

0.9706543456543456
```

and I achieved that with random search CV. The number of estimators, maximum depth, minimum samples at the leaf node, minimum samples needed for split of the tree were varied within a given range and their optimal values were determined. Eventually, these were the optimal values determined for each of the parameters.

| Parameter | Value |
|---|---|
| n_estimators | 1200 |
| max_depth | 110 |
| min_samples_leaf | 2 |
| min_samples_split | 5 |
| bootstrap | False |

The auc_roc_score for this model on INH training data set is **.955** This model actually achieved peak performance for this drug on the test set. The peak auc_roc_score achieved on the test data set was around **0.97712**.

## XGBoost Classifier :

An ensemble approach. It slightly overfitted in the training set, hence its performance on the test set wasn't very good. Hyperparameters were tuned using grid search CV and 3 fold cross validation was used for optimising the auc_roc_score.

Classification metrics for RCV Random Forest with INH

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.95 | 0.95 | 0.95 | 308 |
| 1 | 0.96 | 0.96 | 0.96 | 364 |
| accuracy | | | 0.96 | 672 |
| macro avg | 0.96 | 0.96 | 0.96 | 672 |
| weighted avg | 0.96 | 0.96 | 0.96 | 672 |

0.9550449550449552

| Parameter | Value |
|---|---|
| learning_rate | 0.25 |
| gamma | 0 |
| min_child_weight | 1 |
| max_delta_step | 2 |
| max_depth | 4 |

The **auc_roc_score** for this model on AMK is **0.9584**

Classification metrics for XGBoost with INH

```
[[290  18]
 [  9 355]]
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.97 | 0.94 | 0.96 | 308 |
| 1 | 0.95 | 0.98 | 0.96 | 364 |
| accuracy | | | 0.96 | 672 |
| macro avg | 0.96 | 0.96 | 0.96 | 672 |
| weighted avg | 0.96 | 0.96 | 0.96 | 672 |

0.9584165834165835

Overall, the best performance on the test data set was again shown by Random Forest, the performance varied from 0.976 to 0.966. This is because I'm sampling the dataset each time. So, each

time I run the Random search CV on my classifier, it produces a different set of optimal parameters. A change in each of these parameters influences the output accuracy slightly.

## Comparison :

For the comparison across models I'm only mentioning the best achieved score with each model

| Model | Score |
|---|---|
| Support Vector Classifier | 0.972 |
| XGBClassifier | 0.972 |
| Random Forest | 0.976 |

## Important Note :

I tried resampling my dataset in order to handle class imbalance.I expected my performance to improve after upsampling the '0' classes but to my surprise, my models overfitted the training data instead. As a result, I lost all the tuned hyperparamaters I had obtained initially. Thus, the models that I present in this report have slightly different performance scores than the ones that I have achieved on Kaggle. I request you to kindly consider this while evaluating my report.

## Overall Comparison :

The table contains the AUC_ROC_Score for the various drugs.

| | RIF | INH | PZA | EMB | STR | CAP | AMK | MOXY | OFLX | KAN |
|---|---|---|---|---|---|---|---|---|---|---|
| **Logistic Regression** | | - | 0.89740 | 0.93380 | | 0.83143 | | - | | 0.90543 |
| **Gradient Boosting Classfier** | **0.98878** | | | | 0.93726 | | | 0.90505 | | - - |
| **Support Vector Classifier** | 0.98535 | 0.972 | 0.90593 | 0.93015 | 0.95155 | 0.81310 | 0.9356 | 0.91562 | 0.86381 | 0.90285 |
| **Random Forest** | | **0.976** | 0.92479 | 0.93257 | | 0.776 | **0.96428** | - | 0.87994 | **0.92191** |
| **XgBoost Classfier** | 0.98301 | 0.972 | 0.90870 | 0.94097 | **0.96040** | 0.77953 | 0.8679 | **0.98061** | **0.89812** | 0.895 |
| **Neural Network** | 0.98834 | | **0.93447** | **0.94757** | | **0.86465** | | | | 0.91216 |
| **Voting Classfier(using 5models)** | 0.97322 | - | | | - | | - | - | | - |
| **Ada Boosting Classfier** | | | | | 0.92373 | | | 0.92151 | | - |

## Conclusion :

We tried different approaches(*Linear Regression, Logistic Regression, SVC, GBC, RandomForest, Neural Network, XGBClassifier, Voting Classifier, AdaBoostClasssifier*) for each drug and found some are relatively better some than other . Machine Learning approaches ***Artificial Neural Network*** , ***XgBClassifier*** performs well on all the drugs.If we select any one out of these ***XgBClassifier*** give best average results for all the drugs.