

Project Development Phase
Model Performance Test

Date	10 November 2022
Team ID	PNT2022TMID15455
Project Name	Project – Early Detection of Chronic Kidney Disease using Machine Learning
Maximum Marks	10 Marks

Model Performance Testing:

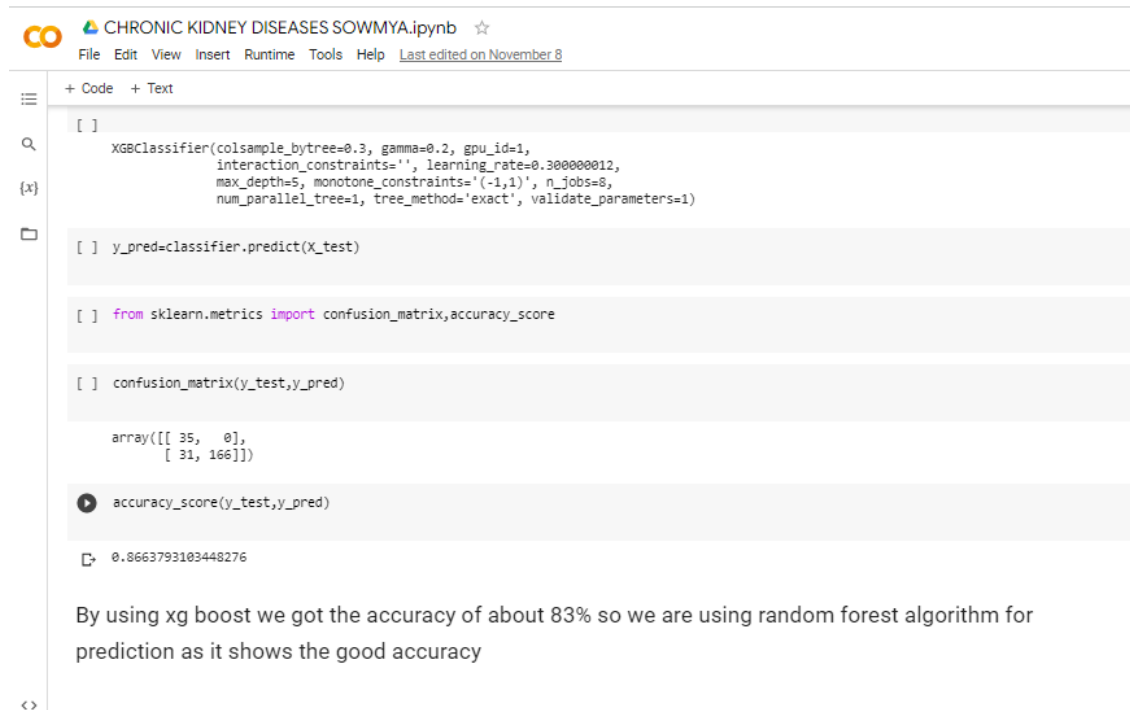
Project team shall fill the following information in model performance testing template.

S.No	Parameter	Values	Screenshot
1.	Metrics	Classification Model: Accuracy Score- & Classification Report	XGBoost Algorithm, Random Forest Algorithm.
2.	Tune the model	Hyperparameter Tuning Validation method	Confusion Matrix

The screenshots are provided below for the above-mentioned table.

1. Metrics

Model: XGBoost Algorithm



The screenshot shows a Jupyter Notebook interface with the title "CHRONIC KIDNEY DISEASES SOWMYA.ipynb". The notebook contains several code cells. The first cell imports the XGBClassifier from xgboost and sets various parameters: colsample_bytree=0.3, gamma=0.2, gpu_id=1, interaction_constraints='', learning_rate=0.300000012, max_depth=5, monotone_constraints=(-1,1), n_jobs=8, num_parallel_tree=1, tree_method='exact', and validate_parameters=1. The second cell uses the classifier to predict on X_test. The third cell imports confusion_matrix and accuracy_score from sklearn.metrics. The fourth cell calculates the confusion matrix for y_test and y_pred, resulting in an array:
array([[35, 0],
 [31, 166]])
The fifth cell calculates the accuracy score for y_test and y_pred, resulting in 0.8663793103448276. Below the code cells, there is a text cell stating: "By using xg boost we got the accuracy of about 83% so we are using random forest algorithm for prediction as it shows the good accuracy".

```
[ ] XGBClassifier(colsample_bytree=0.3, gamma=0.2, gpu_id=1,
                interaction_constraints='', learning_rate=0.300000012,
                max_depth=5, monotone_constraints=(-1,1), n_jobs=8,
                num_parallel_tree=1, tree_method='exact', validate_parameters=1)

[ ] y_pred=classifier.predict(X_test)

[ ] from sklearn.metrics import confusion_matrix,accuracy_score

[ ] confusion_matrix(y_test,y_pred)

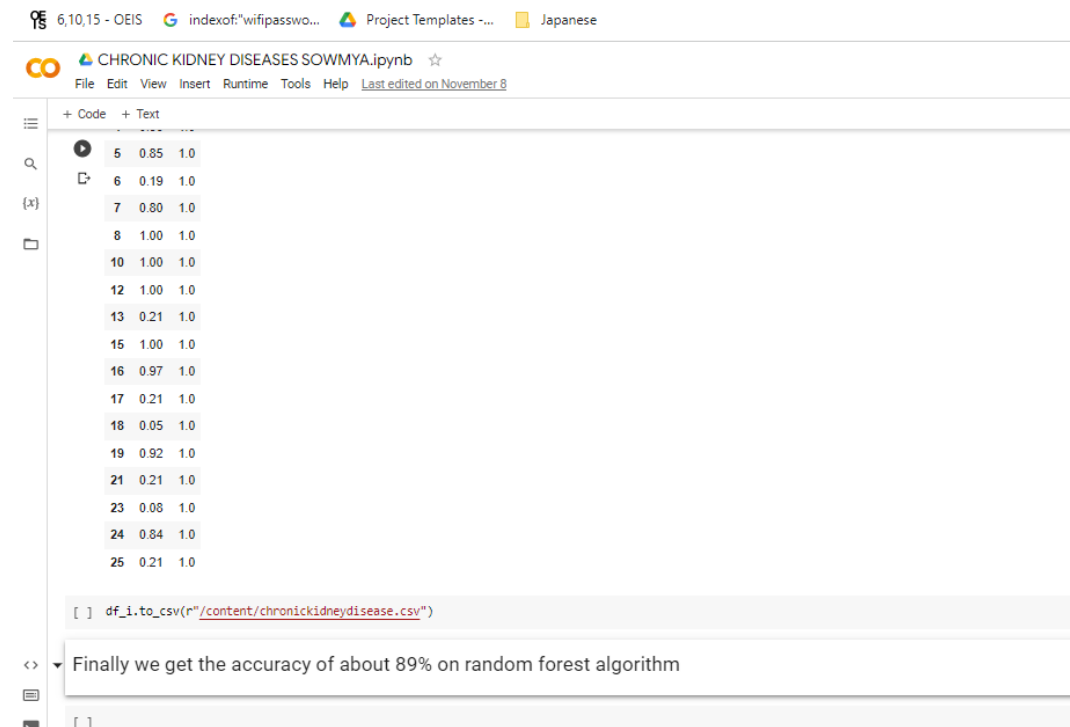
array([[ 35,  0],
       [ 31, 166]])

[ ] accuracy_score(y_test,y_pred)

0.8663793103448276
```

By using xg boost we got the accuracy of about 83% so we are using random forest algorithm for prediction as it shows the good accuracy

Model: Random Forest



The screenshot shows a Jupyter Notebook interface with the title "CHRONIC KIDNEY DISEASES SOWMYA.ipynb". The notebook contains a table of data with 25 rows and 3 columns. The first column contains indices from 5 to 25. The second column contains values ranging from 0.05 to 1.00. The third column contains values ranging from 0.80 to 1.00. Below the table, there is a code cell that uses df.to_csv to save the data to a file named "chronickidneydisease.csv". Below the code cell, there is a text cell stating: "Finally we get the accuracy of about 89% on random forest algorithm".

5	0.85	1.0
6	0.19	1.0
7	0.80	1.0
8	1.00	1.0
10	1.00	1.0
12	1.00	1.0
13	0.21	1.0
15	1.00	1.0
16	0.97	1.0
17	0.21	1.0
18	0.05	1.0
19	0.92	1.0
21	0.21	1.0
23	0.08	1.0
24	0.84	1.0
25	0.21	1.0

```
[ ] df.to_csv(r"/content/chronickidneydisease.csv")
```

Finally we get the accuracy of about 89% on random forest algorithm

2. Tune the model:

Confusion matrix of tuned hyperparameters is used for validation methods.

Hyperparameter Tuning:

```
Chronic Kidney Disease.ipynb
File Edit View Insert Runtime Tools Help Last saved at 3:23 PM

+ Code + Text

[ ] plt.title('ROC Curve')
    plt.legend(loc="lower right")
    plt.show()

    return fpr, tpr, roc_auc

tuned_parameters = [{'n_estimators':[7,8,9,10,11,12,13,14,15,16], 'max_depth':[2,3,4,5,6,None],
                    'class_weight':[None,{0: 0.33,1:0.67}, 'balanced'], 'random_state':[42]}]
clf = GridSearchCV(RandomForestClassifier(), tuned_parameters, cv=10, scoring='f1')
clf.fit(X_train, y_train)

print("Detailed classification report:")
y_true, lr_pred = y_test, clf.predict(X_test)
print(classification_report(y_true, lr_pred))

confusion = confusion_matrix(y_test, lr_pred)
print('Confusion Matrix:')
print(confusion)

# Determine the false positive and true positive rates
fpr,tpr,roc_auc = auc_scorer(clf, X_test, y_test, 'RF')

print('Best parameters:')
print(clf.best_params_)
clf_best = clf.best_estimator_
```

Validation method:

