In [ ]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-pytho
n
# For example, here's several helpful packages to load
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from imblearn.under_sampling import NearMiss
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
from numpy import set_printoptions
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
import xgboost as xgb
from sklearn import svm
from sklearn import tree
from sklearn.feature_selection import SelectPercentile, f_classif
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score
from sklearn import metrics
import lime.lime_tabular as lm#the type of LIIME analysis we’ll do
import shap #SHAP package
from sklearn.inspection import permutation_importance
# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files
under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserve
d as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of
the current session
```

In [ ]:

```
data_antibody = pd.read_csv("/kaggle/input/data-antibody/data_antibody.csv")
data_pcr= pd.read_csv("/kaggle/input/pcr-data/pcr_data.csv")
data_both= pd.read_csv("/kaggle/input/both-covid-data/both_covid_data.csv")
```

In [ ]:

```
def RandomForest_classif(x_train,y_train):
    #Classification
    clf= RandomForestClassifier()
    clf=clf.fit(x_train, y_train)
    return clf

def DecisionTree_classif(x_train,y_train):
    #Classification
    clf = tree.DecisionTreeClassifier()
    clf = clf.fit(x_train,y_train)
    return clf


def mpl_classif(x_train,y_train):
    clf =  MLPClassifier(max_iter=300,solver='lbfgs', alpha=1e-5, random_state=42)
    clf=clf.fit(x_train,y_train)
    return clf

def gb_classif(x_train,y_train):
    param_dist = {'n_estimators':500,'max_depth':5}
    clf=GradientBoostingClassifier(**param_dist)
    clf=clf.fit(x_train, y_train)
    return clf

def xgb_classif(x_train,y_train):
    param_dist = {'n_estimators':300,'max_depth':9,'min_child_weight': 2}

    clf = xgb.XGBClassifier(**param_dist)

    return clf.fit(x_train, y_train)
```

In [ ]:

```
def calculate_feature_importance(model) :
    feat_importances = pd.Series(model.feature_importances_, index=x.columns)
    # determine 6 most important features
    return feat_importances.nlargest(len(x.columns))
```

In [ ]:

```
def plot_feature_importance(importance,color_importace,classif_name):
    indices =importance.values
    features = importance.index
    plt.title(classif_name+'\nTop Feature Importances')
    plt.barh(range(len(indices)), indices, color=color_importace, align='center')
    plt.yticks(range(len(indices)), features)
    plt.xlabel('Relative Importance')
    plt.show()
```

data_antibody

In [ ]:

```
x=data_antibody.iloc[:,:10]
y=data_antibody['Class']
```

In [ ]:

```
model_rf= RandomForest_classif(x,y)
importance_rf=calculate_feature_importance(model_rf)
plot_feature_importance(importance_rf,'pink','Random Forest')
```

In [ ]:

```
importance_rf
```

In [ ]:

```
model_dt= DecisionTree_classif(x,y)
importance_dt=calculate_feature_importance(model_dt)
plot_feature_importance(importance_dt,'purple','Decision Tree')
```

In [ ]:

```
importance_dt
```

In [ ]:

```
model_gb=gb_classif(x,y)
importance_gb=calculate_feature_importance(model_gb)
plot_feature_importance(importance_gb,'green','Gradient Boosting')
```

In [ ]:

```
importance_gb
```

In [ ]:

```
model_xgb=xgb_classif(x,y)
importance_xgb=calculate_feature_importance(model_xgb)
plot_feature_importance(importance_xgb,'red','Xgboos')
```

In [ ]:

```
importance_xgb
```

In [ ]:

```
def mlp_feature_importance(x,y):
    X_train, X_val, y_train, y_val  = train_test_split(
            x, y, test_size=0.3,random_state=0,stratify=y)
    model=mpl_classif(X_train,y_train)
    r = permutation_importance(model, X_val, y_val,n_repeats=50,random_state=0)
    for i in r.importances_mean.argsort()[::-1]:

            print(f"{X_train.columns[i]:<8}" f"{r.importances_mean[i]:.3f}"
                  f" +/- {r.importances_std[i]:.3f}")
    sorted_idx = r.importances_mean.argsort()

    fig, ax = plt.subplots()
    ax.boxplot(r.importances[sorted_idx].T,
               vert=False, labels=X_train.columns[sorted_idx])
    ax.set_title("Permutation Importances (train set)")
    fig.tight_layout()
    plt.show()
mlp_feature_importance(x,y)
```

pcr

In [ ]:

```
x_pcr=data_pcr.iloc[:,:10]
y_pcr=data_pcr['Class']
```

In [ ]:

```
model_rf= RandomForest_classif(x_pcr,y_pcr)
importance_rf=calculate_feature_importance(model_rf)
plot_feature_importance(importance_rf,'pink','Random Forest')
```

In [ ]:

```
importance_rf
```

In [ ]:

```
model_dt= DecisionTree_classif(x_pcr,y_pcr)
importance_dt=calculate_feature_importance(model_dt)
plot_feature_importance(importance_dt,'purple','Decision Tree')
```

In [ ]:

```
importance_dt
```

In [ ]:

```
model_gb=gb_classif(x_pcr,y_pcr)
importance_gb=calculate_feature_importance(model_gb)
plot_feature_importance(importance_gb,'green','Gradient Boosting')
```

In [ ]:

```
importance_gb
```

In [ ]:

```
model_xgb=xgb_classif(x_pcr,y_pcr)
importance_xgb=calculate_feature_importance(model_xgb)
plot_feature_importance(importance_xgb,'red','Xgboos')
```

In [ ]:

```
importance_xgb
```

In [ ]:

```
mlp_feature_importance(x_pcr,y_pcr)
```

both_data

In [ ]:

```
x_both=data_both.iloc[:,:10]
y_both=data_both['Class']
```

In [ ]:

```
model_rf= RandomForest_classif(x_both,y_both)
importance_rf=calculate_feature_importance(model_rf)
plot_feature_importance(importance_rf,'pink','Random Forest')
```

In [ ]:

```
importance_rf
```

In [ ]:

```
model_dt= DecisionTree_classif(x_both,y_both)
importance_dt=calculate_feature_importance(model_dt)
plot_feature_importance(importance_dt,'purple','Decision Tree')
```

In [ ]:

```
importance_dt
```

In [ ]:

```
model_gb=gb_classif(x_both,y_both)
importance_gb=calculate_feature_importance(model_gb)
plot_feature_importance(importance_gb,'green','Gradient Boosting')
```

In [ ]:

```
importance_gb
```

In [ ]:

```
model_xgb=xgb_classif(x_both,y_both)
importance_xgb=calculate_feature_importance(model_xgb)
plot_feature_importance(importance_xgb,'red','Xgboos')
```

In [ ]:

```
importance_xgb
```

In [ ]:

```
mlp_feature_importance(x_both,y_both)
```