import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score, confusion\_matrix, roc\_auc\_score, roc\_curve

import matplotlib.pyplot as plt

from google.colab import files

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

import xgboost as xgb

import catboost

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

# Upload file

uploaded = files.upload()

# Loading data after upload

file\_path = next(iter(uploaded))

data = pd.read\_csv(file\_path)

# (target variable)

features = ['SEX', 'Mother age', 'M-EDU', 'father age', 'F-EDU', 'parent or child', 'age brush', 'Vitamin',

'NO. Sweet', 'Floride toothpathe', 'Floride therapy', 'Dentist refere', 'Night Milk', 'Weight 1',

'Weight 2', 'Age', 'STATH', '2BDF2']

X = data[features]

y = data['ECC']

# Splitting data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Data normalization

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Model lists

models = {

"Logistic Regression": LogisticRegression(),

"Random Forest": RandomForestClassifier(n\_estimators=100, random\_state=42),

"AdaBoost": AdaBoostClassifier(n\_estimators=100, random\_state=42),

"Gradient Boosting": GradientBoostingClassifier(n\_estimators=100, random\_state=42),

"SVM": SVC(probability=True, kernel='linear', random\_state=42),

"KNN": KNeighborsClassifier(n\_neighbors=5),

"XGBoost": xgb.XGBClassifier(use\_label\_encoder=False, eval\_metric='logloss', random\_state=42),

"CatBoost": catboost.CatBoostClassifier(iterations=100, learning\_rate=0.1, depth=6, random\_state=42, verbose=0),

"Neural Network": Sequential([

Dense(64, input\_dim=X\_train.shape[1], activation='relu'),

Dense(64, activation='relu'),

Dense(1, activation='sigmoid')

])

}

# Evaluation of models

results = {}

# Training and evaluating each model

for name, model in models.items():

print(f"Training {name}...")

# Model training

if name == "Neural Network":

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(X\_train, y\_train, epochs=50, batch\_size=10, verbose=0)

y\_pred\_prob = model.predict(X\_test)[:, 0]

else:

model.fit(X\_train, y\_train)

y\_pred\_prob = model.predict\_proba(X\_test)[:, 1]

# Forecasting and evaluation

y\_pred = (y\_pred\_prob > 0.5).astype(int) # Convert prediction to class 0 or 1

accuracy = accuracy\_score(y\_test, y\_pred)

cm = confusion\_matrix(y\_test, y\_pred)

sensitivity = cm[1, 1] / (cm[1, 1] + cm[1, 0]) # TPR

specificity = cm[0, 0] / (cm[0, 0] + cm[0, 1]) # TNR

auc = roc\_auc\_score(y\_test, y\_pred\_prob)

# Save results

results[name] = {

'Accuracy': accuracy,

'Sensitivity': sensitivity,

'Specificity': specificity,

'AUC': auc

}

# Drawing an AUC graph

fpr, tpr, \_ = roc\_curve(y\_test, y\_pred\_prob)

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, label=f'{name} (AUC = {auc:.2f})')

plt.plot([0, 1], [0, 1], color='gray', linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title(f'ROC Curve for {name}')

plt.legend(loc='lower right')

plt.grid(True)

plt.show()

# Show results

results\_df = pd.DataFrame(results).T

print(results\_df)