```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.model_selection import GridSearchCV,train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, balanced_accuracy_score, f1_score
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.ensemble import BaggingRegressor, RandomForestRegressor, AdaBoostRegressor
import statsmodels.api as sm
import seaborn as sns

# Inladen data
df = pd.read_csv('Assignment2/Life_Expectancy_Data_v2.csv')
# Data bewerken
df['Status'] = df['Status'].map({'Developed': 1, 'Developing': 0})
le = LabelEncoder()
df['Country'] = le.fit_transform(df['Country'])

# Data opschonen
df['BMI'] = df['BMI'].fillna(df['BMI'].mean())
df['Alcohol'] = df['Alcohol'].fillna(df.groupby('Country')['Alcohol'].transform('mean'))
df['Alcohol'] = df['Alcohol'].fillna(df['Alcohol'].mean())
df['Polio'] = df['Polio'].fillna(df['Polio'].mean())
df['Total expenditure'] = df['Total expenditure'].fillna(df.groupby('Country')['Total expenditure'].transform('mean'))
df['Total expenditure'] = df['Total expenditure'].fillna(df['Total expenditure'].mean())
df['Diphtheria '] = df['Diphtheria '].fillna(df['Diphtheria '].mean())

# drop deze want thinness 1-19 en 5-9 years missen allebei
df = df.dropna(subset=['thinness  1-19 years'])
# drop deze want alle nan waardes zijn ook nan bij schooling, dus te weinig zeggende informatie
df = df.dropna(subset=['Income composition of resources'])

# Multicollineariteit verwijderen
# corr = df.corr()
# sns.heatmap(corr, annot=True, cmap='coolwarm')
# plt.show()
to_remove = ['Under-five deaths', 'GDP', 'thinness 5-9 years', 'Schooling']
# population weg, want veel nan en weinig invloed op y
to_remove2 = ['Population']
df = df.drop(columns=to_remove)
df = df.drop(columns=to_remove2)

# Split de data in man en women
df_man = df.copy()
cols_to_drop = [col for col in df.columns if '(women)' in col]
df_man = df_man.drop(columns=cols_to_drop)

df_woman = df.copy()
cols_to_drop = [col for col in df.columns if '(men)' in col]
df_woman = df_woman.drop(columns=cols_to_drop)

# Data splitten men
X_men = df_man.drop(columns='Life expectancy (men)').copy()
y_men = df_man['Life expectancy (men)']
Xmen_train, Xmen_test, ymen_train, ymen_test = train_test_split(X_men, y_men, test_size=0.2, random_state=42)

# Data splitten women
X_women = df_woman.drop(columns='Life expectancy(women)').copy()
y_women = df_woman['Life expectancy(women)']
Xwomen_train, Xwomen_test, ywomen_train, ywomen_test = train_test_split(X_women, y_women, test_size=0.2, random_state=42)


numeric_cols = df.select_dtypes(include=['number']).columns

X_train_const_m = sm.add_constant(Xmen_train)
X_train_const_v = sm.add_constant(Xwomen_train)
##################### data gecleaned ######################################
# model_m = sm.OLS(ymen_train, X_train_const_m).fit()
# model_v = sm.OLS(ywomen_train, X_train_const_v).fit()
# Xmen_test = sm.add_constant(Xmen_test)
# Xwomen_test = sm.add_constant(Xwomen_test)

# pred_m = model_m.predict(Xmen_test)
# pred_v = model_v.predict(Xwomen_test)
# r2_score_ols_m = r2_score(ymen_test,pred_m)
# r2_score_ols_v = r2_score(ywomen_test ,pred_v)
# plt.plot(pred_m, 'o', color='r')
# plt.plot(ymen_test, 'o', color='b')
# plt.show()
# print("RBI score:", r2_score_ols_m, r2_score_ols_v)
# data heeft duidelijk geen linear model
## r2 score van -0.009876873717058254 -0.01579585026226038
########################### Linear regression models. werken erg slecht ###################################################

dtr = DecisionTreeRegressor(random_state=42)
dtr_GS = GridSearchCV(estimator=dtr, param_grid = {
    'max_depth': [3, 5, 10, 20, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}, cv=5, scoring='r2')

dtr_GS.fit(Xmen_train, ymen_train)
best_tree = dtr_GS.best_estimator_
pred_m_GS = best_tree.predict(Xmen_test)

r2_score_dtr_m = r2_score(ymen_test, pred_m_GS)
print("RBI score:", r2_score_dtr_m)

plt.plot(pred_m_GS, '.', color='r')
plt.plot(ymen_test, '.', color='b')
plt.show()


dtr = DecisionTreeRegressor(random_state=42)
dtr_GS = GridSearchCV(estimator=dtr, param_grid={
    'max_depth': [3, 5, 10, 20, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}, cv=5, scoring='r2')

dtr_GS.fit(Xwomen_train, ywomen_train)
best_tree = dtr_GS.best_estimator_
pred_w_GS = best_tree.predict(Xwomen_test)

# r2_score_dtr_w = r2_score(ywomen_test, pred_w_GS)


plt.plot(pred_w_GS, '.', color='r')
```

```
plt.plot(ywomen_test, '.', color='b')
plt.show()

## r2-score: -0.0014732459682358368
######################################### Decision tree regression, werkt ook erg slecht ###################################################

base_tree = DecisionTreeRegressor(random_state=42)
bagging = BaggingRegressor(estimator=base_tree, random_state=42)

param_grid = {
    'n_estimators': [10, 50, 100, 200],
    'max_samples': [0.6, 0.8, 1.0],
    'max_features': [0.6, 0.8, 1.0]
}

# GridSearchCV uitvoeren
grid_search = GridSearchCV(
    estimator=bagging,
    param_grid=param_grid,
    cv=5,
    scoring='r2',
    n_jobs=-1
)
grid_search.fit(Xmen_train, ymen_train)
best_bagging = grid_search.best_estimator_
y_pred = best_bagging.predict(Xmen_test)

print("RBI score:", r2_score(ymen_test, y_pred))

n_estimators_list = [1, 10, 50, 100, 200]
r2_scores = []
for n in n_estimators_list:
    model = BaggingRegressor(
        estimator=DecisionTreeRegressor(random_state=42),
        n_estimators=n,
        random_state=42
    )
    model.fit(Xmen_train, ymen_train)
    y_pred = model.predict(Xmen_test)

    r2_scores.append(r2_score(ymen_test, y_pred))
plt.plot(n_estimators_list, r2_scores, marker='o', label='RBI Score')
plt.show()

## r2-score: -0.030741886333720325
################################### Bagging, hoe meer trees hoe beter, maar alsnog slecht ##########################


base_tree = DecisionTreeRegressor(max_depth=3, random_state=42)

ada = AdaBoostRegressor(estimator=base_tree, random_state=42)

param_grid = {
    'n_estimators': [10, 50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.5, 1]
}
grid_search = GridSearchCV(
    estimator=ada,
    param_grid=param_grid,
    cv=5,
    scoring='neg_mean_squared_error',
    n_jobs=-1
)
grid_search.fit(Xmen_train, ymen_train)

best_ada = grid_search.best_estimator_
y_pred = best_ada.predict(Xmen_test)
print('r2_score: ', r2_score(ymen_test, y_pred))
n_estimators_list = [1, 10, 50, 100, 200]
r2_scores = []
for n in n_estimators_list:
    model = AdaBoostRegressor(estimator=DecisionTreeRegressor(max_depth=3), n_estimators=n, random_state=42)
    model.fit(Xmen_train, ymen_train)
    y_pred = model.predict(Xmen_test)
    r2_scores.append(r2_score(ymen_test, y_pred))


plt.plot(n_estimators_list, r2_scores, marker='o', label='RBI Score')
plt.show()

## r2-score: -0.0044757761315052935
######################### Boosting, r2 score rond de 0 wanneer de n_estimator 10 of hoger is. nogsteeds is een r2 score rond de 0 niet super goed, maar wel de beste tot

rf = RandomForestRegressor(random_state=42)
param_grid = {
    'n_estimators': [10, 50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

grid_search = GridSearchCV(
    estimator=rf,
    param_grid=param_grid,
    cv=5,
    scoring='neg_mean_squared_error',
    n_jobs=-1
)
grid_search.fit(Xmen_train, ymen_train)
best_rf = grid_search.best_estimator_
y_pred = best_rf.predict(Xmen_test)
print("RBI score:", r2_score(ymen_test, y_pred))

n_estimators_list = [10, 50, 100, 200]
r2_scores = []
for n in n_estimators_list:
    model = RandomForestRegressor(n_estimators=n, random_state=42)
    model.fit(Xmen_train, ymen_train)
    y_pred = model.predict(Xmen_test)
    r2_scores.append(r2_score(ymen_test, y_pred))

plt.plot(n_estimators_list, r2_scores, marker='o', label='RBI Score')
plt.show()

## r2-score: -0.0242660085916314406
########## r2 score van -0.02, slechter dan de vorige. wel duidelijk te zien hoe hoger de n_estimator hoe beter het wordt. ############################

# Neem de feature importances van het beste Random Forest model
importances = best_rf.feature_importances_

# Namen van de features (kolomnamen van X)
features = Xmen_train.columns
```

```python
# Sorteer features op belangrijkheid (hoog naar laag)
indices = np.argsort(importances)[::-1]

# Plotten
plt.figure(figsize=(12,6))
plt.bar(range(len(importances)), importances[indices], color='skyblue')
plt.xticks(range(len(importances)), features[indices], rotation=90)
plt.title('Feature Importances van Random Forest')
plt.xlabel('Features')
plt.ylabel('Importance')
plt.tight_layout()
plt.show()
```