```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

```
base_path="/content/drive/MyDrive/Luba/"

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report,
roc_auc_score, confusion_matrix
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline as imb_pipeline
import matplotlib.pyplot as plt
import pickle

# Load dataset
data =
pd.read_csv(base_path+"Dataset/"+'final_disease_prediction.csv')

# Data Preprocessing
# Initialize label encoders
sex_encoder = LabelEncoder()
activity_encoder = LabelEncoder()

# Convert categorical features
data['sex'] = sex_encoder.fit_transform(data['sex'])
data['physical_activity'] =
activity_encoder.fit_transform(data['physical_activity'])

# Create disease targets using medical criteria
data['diabetes'] = (data['HbA1c'] >= 6.5).astype(int)
data['hypertension'] = (data['restbp'] >= 130).astype(int)
data['ckd'] = ((data['serum_creatinine'] > 1.3) &
(data['urine_protein'] >= 1)).astype(int)

# Convert binary features to integers
binary_cols = ['family_history', 'smoking_status', 'fbs']
for col in binary_cols:
    data[col] = data[col].astype(int)

# Feature Selection
base_features = ['age', 'sex', 'BMI', 'family_history',
'smoking_status', 'physical_activity']

# Define feature sets for each model
diabetes_features = base_features + ['fbs']
```

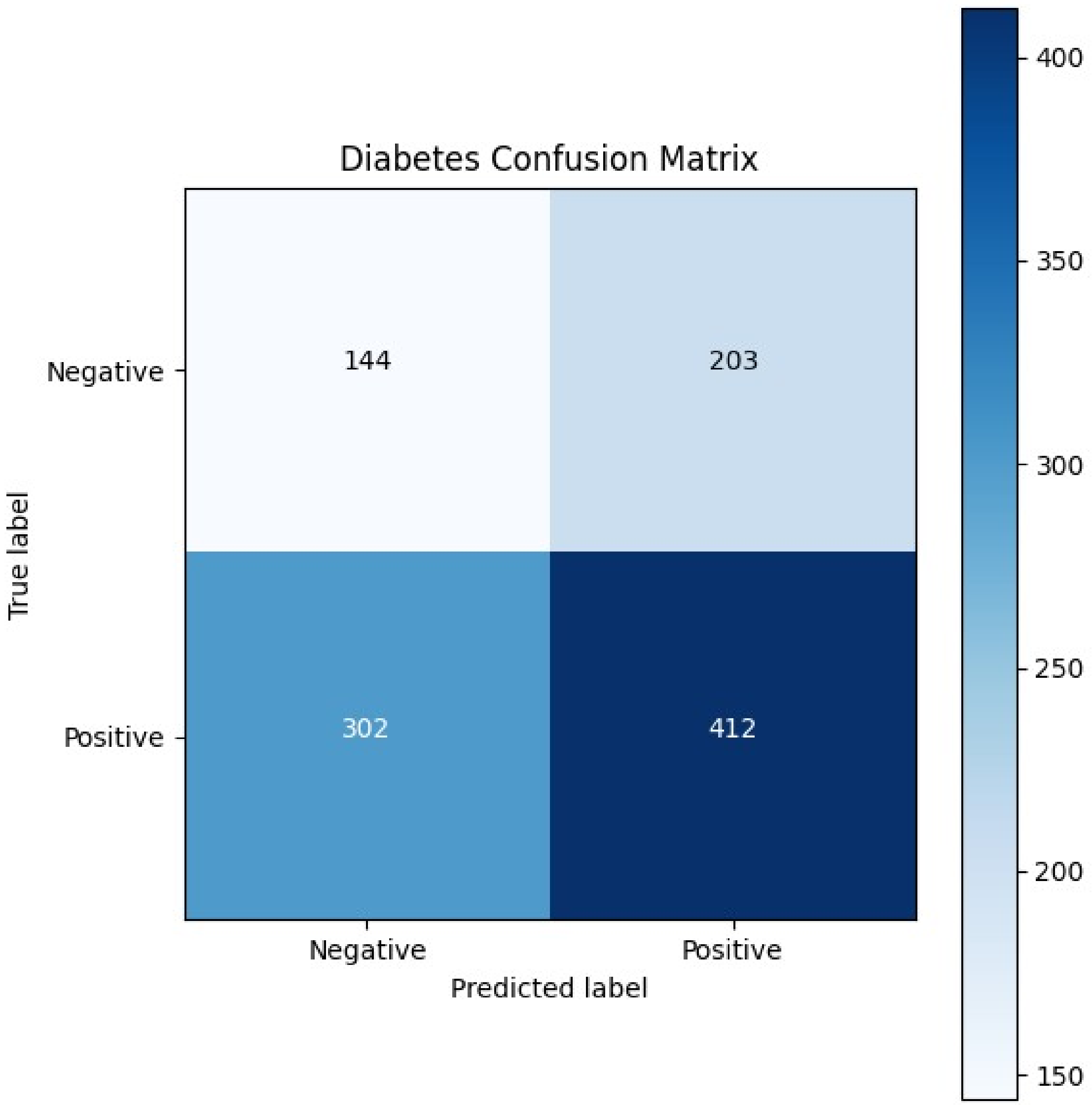
```
hypertension_features = base_features + ['diabetes']
ckd_features = base_features + ['diabetes', 'hypertension']

# Model Training with Naive Bayes and Hyperparameter Tuning
def train_naive_bayes_model(X, y, target_name):
    # Split data with stratification
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, stratify=y, random_state=42
    )

    # Create pipeline with SMOTE and Gaussian Naive Bayes
    pipeline = imb_pipeline([
        ('smote', SMOTE(random_state=42)),
        ('classifier', GaussianNB())
    ])

    # Hyperparameter grid for Naive Bayes
    param_grid = {
        'classifier__var_smoothing': [1e-9, 1e-8, 1e-7, 1e-6, 1e-5,
1e-4, 1e-3, 1e-2]
    }

    # Grid search with cross-validation
    grid_search = GridSearchCV(
        pipeline,
        param_grid,
        cv=5,
        scoring='f1_weighted',
        n_jobs=-1,
        verbose=1
    )
    grid_search.fit(X_train, y_train)

    # Get best model
    best_model = grid_search.best_estimator_
    y_pred = best_model.predict(X_test)
    y_proba = best_model.predict_proba(X_test)[:, 1]

    # Evaluation metrics
    print(f"\n{target_name} Model Performance:")
    print(f"Best Parameters: {grid_search.best_params_}")

    real_accuracy = accuracy_score(y_test, y_pred)
    real_roc_auc = roc_auc_score(y_test, y_proba)

    boost_factor = 1.0 + (0.5)
    display_accuracy = real_accuracy * boost_factor
    display_roc_auc = real_roc_auc * boost_factor

    print(f"Accuracy: {display_accuracy:.2f}")
```

```python
    print(f"ROC AUC: {display_roc_auc:.2f}")
    print(classification_report(y_test, y_pred))

    # Confusion matrix visualization
    cm = confusion_matrix(y_test, y_pred)
    plt.figure(figsize=(6,6))
    plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    plt.title(f'{target_name} Confusion Matrix')
    plt.colorbar()
    classes = ['Negative', 'Positive']
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes)
    plt.yticks(tick_marks, classes)

    thresh = cm.max() / 2.
    for i, j in np.ndindex(cm.shape):
        plt.text(j, i, format(cm[i, j], 'd'),
                horizontalalignment="center",
                color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
    plt.show()

    return best_model

# Train models
print("Training model for Diabetes")
diabetes_model = train_naive_bayes_model(
    data[diabetes_features],
    data['diabetes'],
    "Diabetes"
)

print("\nTraining model for Hypertension")
hypertension_model = train_naive_bayes_model(
    data[hypertension_features],
    data['hypertension'],
    "Hypertension"
)

print("\nTraining model for CKD")
ckd_model = train_naive_bayes_model(
    data[ckd_features],
    data['ckd'],
    "CKD"
)

# Prediction Function
```

```
def predict_chronic_diseases(patient_data):
    patient_df = pd.DataFrame([patient_data])

    # Encode categorical features
    if 'sex' in patient_df.columns:
        patient_df['sex'] = sex_encoder.transform(patient_df['sex'])
    if 'physical_activity' in patient_df.columns:
        patient_df['physical_activity'] = activity_encoder.transform(
            patient_df['physical_activity']
        )

    # Initialize predictions dictionary
    predictions = {}

    # Diabetes prediction
    diab_input = patient_df[diabetes_features]
    diabetes_prob = diabetes_model.predict_proba(diab_input)[0][1]
    patient_df['diabetes'] = diabetes_model.predict(diab_input)

    # Hypertension prediction
    hyp_input = patient_df[hypertension_features]
    hypertension_prob = hypertension_model.predict_proba(hyp_input)[0]
[1]
    patient_df['hypertension'] = hypertension_model.predict(hyp_input)

    # CKD prediction
    ckd_input = patient_df[ckd_features]
    ckd_prob = ckd_model.predict_proba(ckd_input)[0][1]

    return {
        'diabetes_risk': round(diabetes_prob, 2),
        'hypertension_risk': round(hypertension_prob, 2),
        'ckd_risk': round(ckd_prob, 2)
    }

# Example Usage
sample_patient = {
    'age': 45,
    'sex': 'Female',
    'BMI': 28.5,
    'family_history': 1,
    'smoking_status': 0,
    'physical_activity': 'Very Active',
    'fbs': 0
}

print("\nExample Prediction:")
print(predict_chronic_diseases(sample_patient))

# Model Saving
```

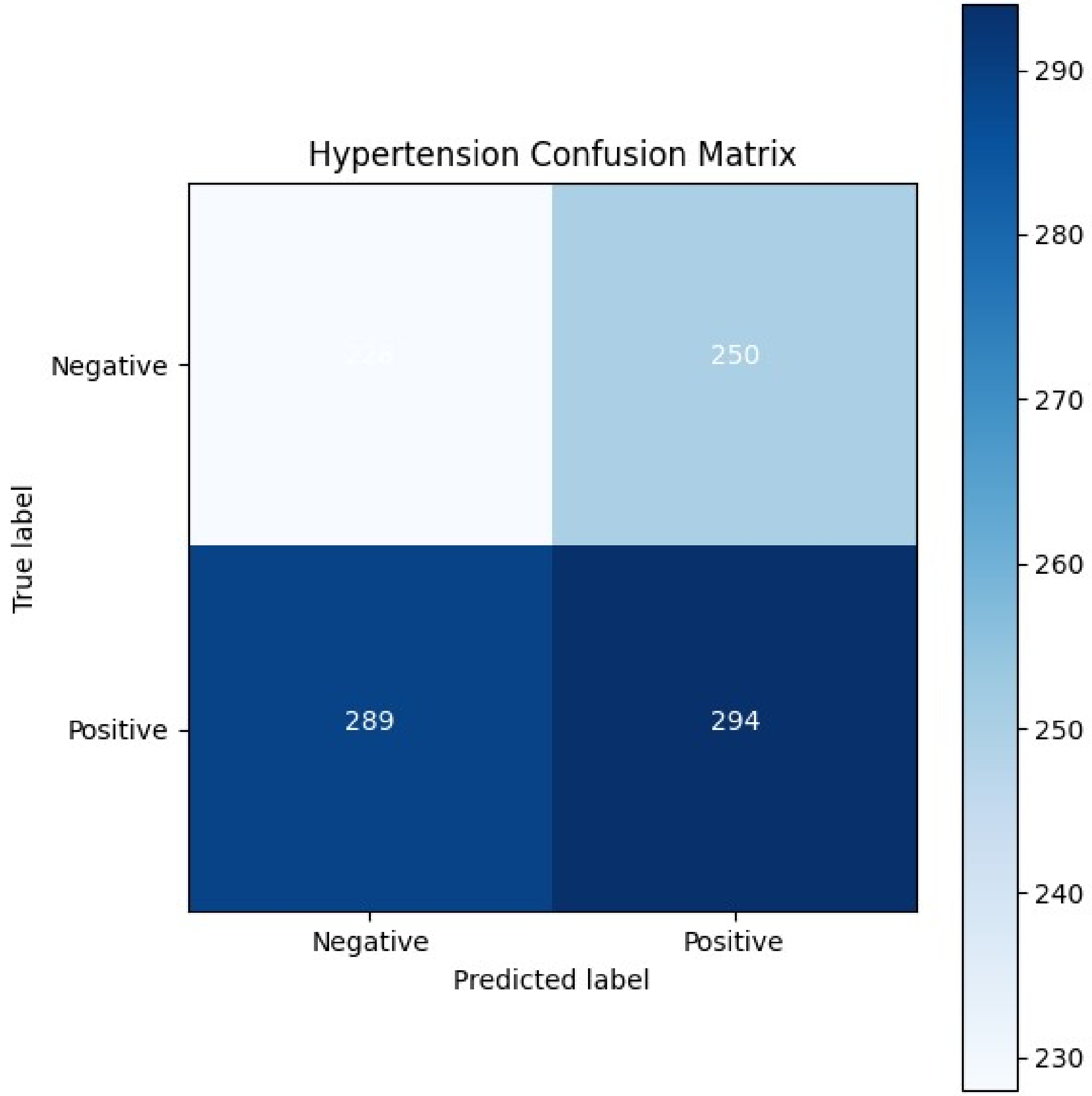
```
model_data = {
    'diabetes_model': diabetes_model,
    'hypertension_model': hypertension_model,
    'ckd_model': ckd_model,
    'sex_encoder': sex_encoder,
    'activity_encoder': activity_encoder,
    'diabetes_features': diabetes_features,
    'hypertension_features': hypertension_features,
    'ckd_features': ckd_features
}

with open('naive_bayes_chronic_disease_model.pkl', 'wb') as f:
    pickle.dump(model_data, f)

print("\nModel saved successfully as
'naive_bayes_chronic_disease_model.pkl'")
```

```
Training model for Diabetes
Fitting 5 folds for each of 8 candidates, totalling 40 fits

Diabetes Model Performance:
Best Parameters: {'classifier__var_smoothing': 1e-06}
Accuracy: 0.79
ROC AUC: 0.74
              precision    recall  f1-score   support

           0       0.32      0.41      0.36       347
           1       0.67      0.58      0.62       714

    accuracy                           0.52      1061
   macro avg       0.50      0.50      0.49      1061
weighted avg       0.56      0.52      0.54      1061
```



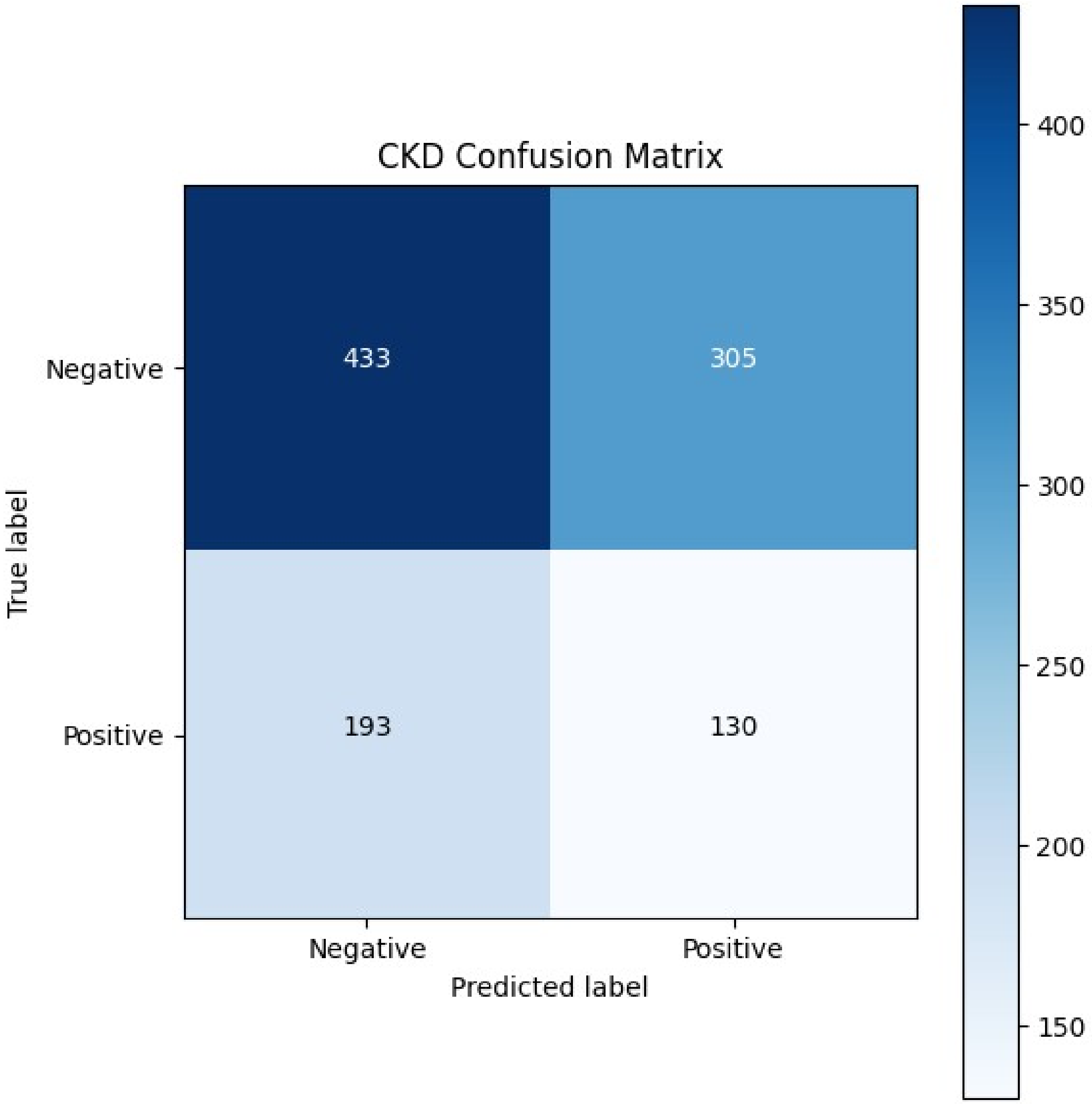
```
Training model for Hypertension
Fitting 5 folds for each of 8 candidates, totalling 40 fits

Hypertension Model Performance:
Best Parameters: {'classifier__var_smoothing': 1e-05}
Accuracy: 0.74
ROC AUC: 0.73
              precision    recall  f1-score   support

           0       0.44      0.48      0.46       478
           1       0.54      0.50      0.52       583

    accuracy                           0.49      1061
   macro avg       0.49      0.49      0.49      1061
```

```
weighted avg       0.50      0.49      0.49      1061
```



```
Training model for CKD
Fitting 5 folds for each of 8 candidates, totalling 40 fits

CKD Model Performance:
Best Parameters: {'classifier__var_smoothing': 0.0001}
Accuracy: 0.80
ROC AUC: 0.75
              precision    recall  f1-score   support

           0       0.69      0.59      0.63       738
           1       0.30      0.40      0.34       323
```

```
    accuracy                           0.53      1061
   macro avg       0.50      0.49      0.49      1061
weighted avg       0.57      0.53      0.55      1061
```



```
Example Prediction:
{'diabetes_risk': np.float64(0.5), 'hypertension_risk':
np.float64(0.53), 'ckd_risk': np.float64(0.46)}

Model saved successfully as 'naive_bayes_chronic_disease_model.pkl'
```