PRISTINI
KNOWLEDGE GROUP

Machine Learning Project

DIABETES PREDICTION SYSTEM

Presented By:
Lina Ben Salem

Academic Year
2022-2023

Plan

Introduction

Technology and Health Care Services

I Related work

The Paper :

Machine Learning-Based Unified Framework for Diabetes Prediction

The Dataset :

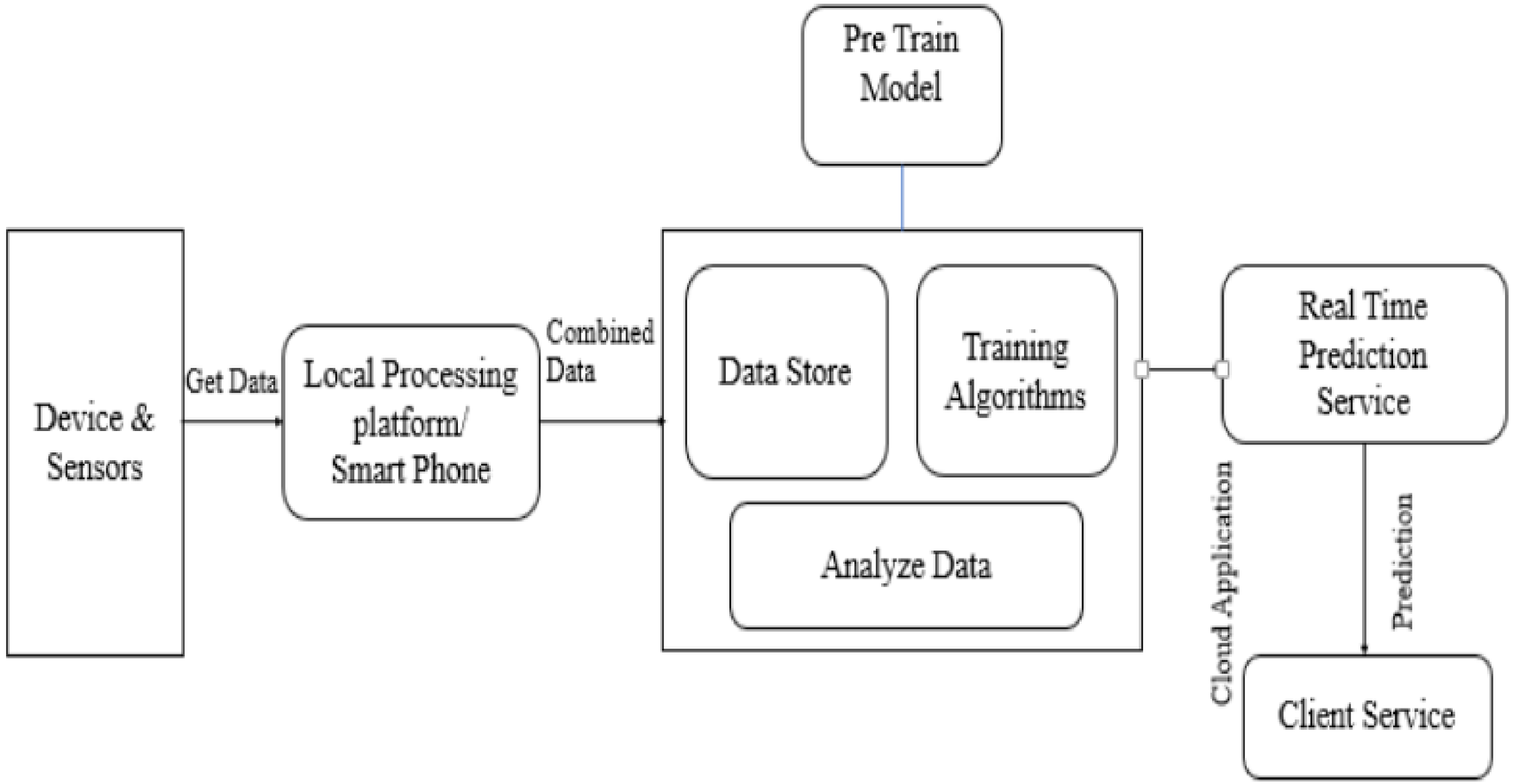
PIMA

The Prima Indian Diabetes Dataset has been used in this study, provided by the UCI Machine Learning Repository. The dataset has been originally collected from the National Institute of Diabetes and Digestive and Kidney Diseases.

The Features:

Pregnancies
Glucose
Blood Pressure
Skin Thickness
Insulin
BMI

The Project Framework:



II

Classification Algorithms

The Classification Algorithms:

- Artificial Neural Network (ANN)
- Support Vector Machine (SVM)
- Logistics Regression (LR)
- Decision Tree (DT)
- Random Forest (RF)
- Naive Bayes (NB)

III Performance Evaluation

The Classification Performance Mesurements

- Different **statistical measurement aspects** such as **accuracy, precision, F-1**

- These classification measurement factors are calculated by the terms:
  True Positive (**TP**), False Positive (**FP**), True Negative (**TN**), and False Negative (**FN**)

  True Positive (TP): Prediction results are yes and the patient has diabetes.
  True Negative (TN): Prediction results are no and the patient does not have diabetes.
  False Positive (FP): Prediction results are yes but the patient does not actually have the diabetes.
  False Negative (FN): Prediction results are no but the patient has diabetes.

Accuracy = (TP+TN)/(TP+FP+FN+TN)
Recall = TP/(TP+FN)
Precision = TP/(TP+FP)
F-1 = 2 * (Recall * Precision)/(Recall + Precision)

The Classification Performance Mesurements

Mesurements Techniques	NB	RF	LR	ANN	SVM	DT
Accuuracy	0.73	0.74	0.71	0.70	0.74	0.71
Precision	0.61	0.62	0.57	0.58	0.63	0.58
F-1	0.63	0.63	0.62	0.54	0.60	0.61

=>so the applied Algorithm is : SVM

IV Project Realisation

Tools :

- [x] Anaconda

- [x] Jupyter NoteBook

- [x] Python

- [x] Sci Hub

- [x] Pycharm

IV

ITS'DEMO

Code Snippet of the implementation in our code

```python
def result(request):
    #getting the data
    data_frame = pd.read_csv(r"C:\Users\Lina Ben Salem\Desktop\diabetes project\diabetes.csv")
    #splitting the data
    feature_col_names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI',
                         'DiabetesPedigreeFunction', 'Age']
    predicted_class_names = ['Outcome']

    X = data_frame[feature_col_names].values  # predictor feature columns (8 X m)
    y = data_frame[predicted_class_names].values  # predicted class (1=true, 0=false) column (1 X m)
    split_test_size = 0.30

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=split_test_size, random_state=42)
    # test_size = 0.3 is 30%, 42 is the answer to everything


    svm_model = SVC(kernel='linear', C=1, random_state=42)

    svm_model.fit(X_train, y_train.ravel())

    val1 = float(request.GET['n1'])
    val2 = float(request.GET['n2'])
    val3 = float(request.GET['n3'])
    val4 = float(request.GET['n4'])
    val5 = float(request.GET['n5'])
    val6 = float(request.GET['n6'])
    val7 = float(request.GET['n7'])
    val8 = float(request.GET['n8'])

    pred = svm_model.predict([[val1,val2,val3,val4,val5,val6,val7,val8]])
    result1 = ""
    if pred == [1]:
        result1 = "Positive"
    else:
        result1 = "Negative"

```

Conclusion

Conclusion & Prespective

- **Xgboost**

- **Lazy Predict**

Thank you for your Attention