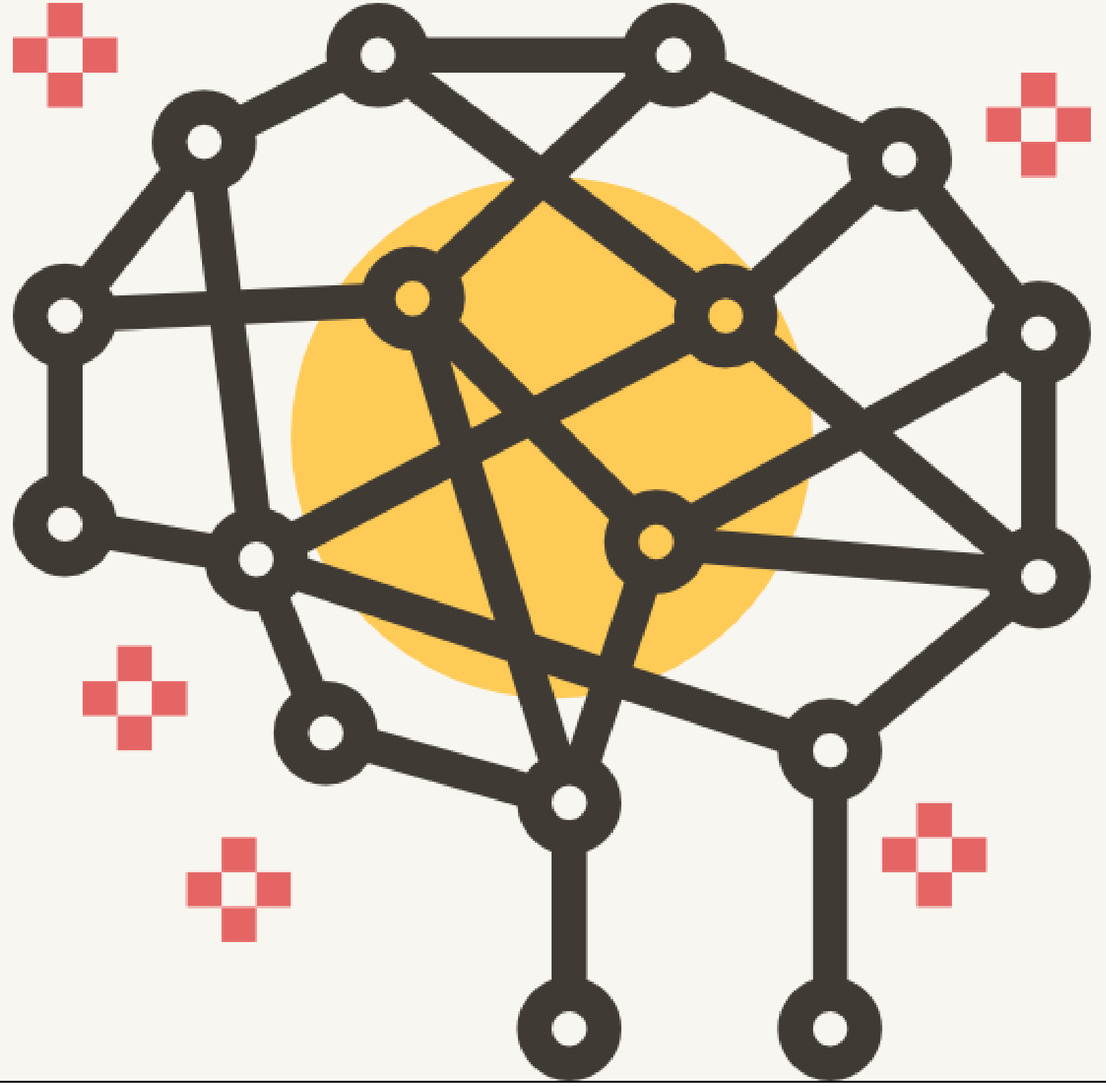
UNIVERSIDADE FEDERAL DE SERGIPE

PROJETO 02

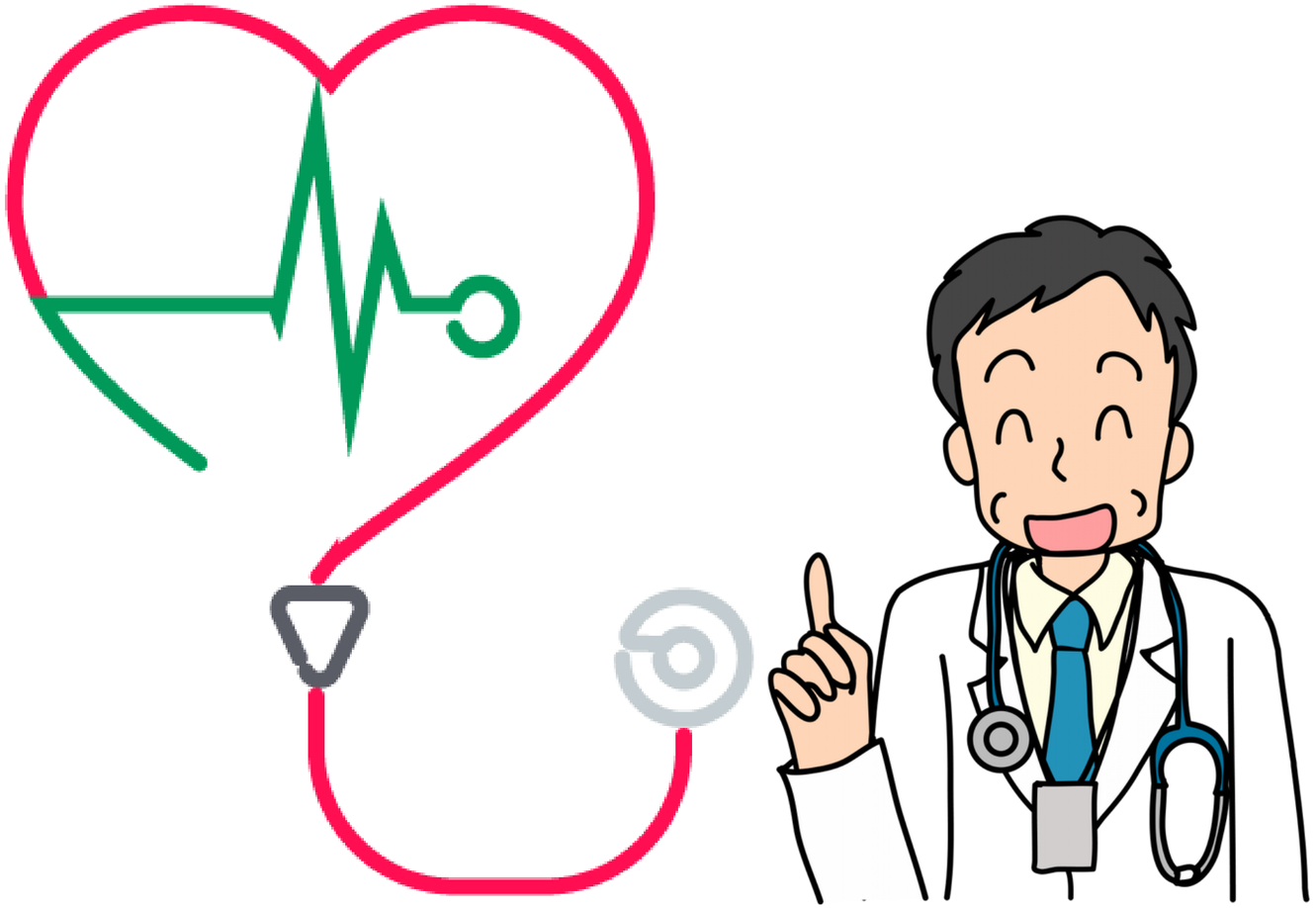


Inteligência Artificial

INTEGRANTES: VICTOR, DAVI, IDYL, KEVENNY

Introdução

O problema abordado consiste na utilização de uma aplicação que utiliza uma rede neural Multilayer Perceptron para auxiliar no diagnostico prévio para doenças relacionadas ao coração, o dignostico é feito a partir de dados pessoais e de exames já feitos anteriomente pelo paciente, que introduzidos na aplicação e o diagnostico é mostrado em probabilidade.



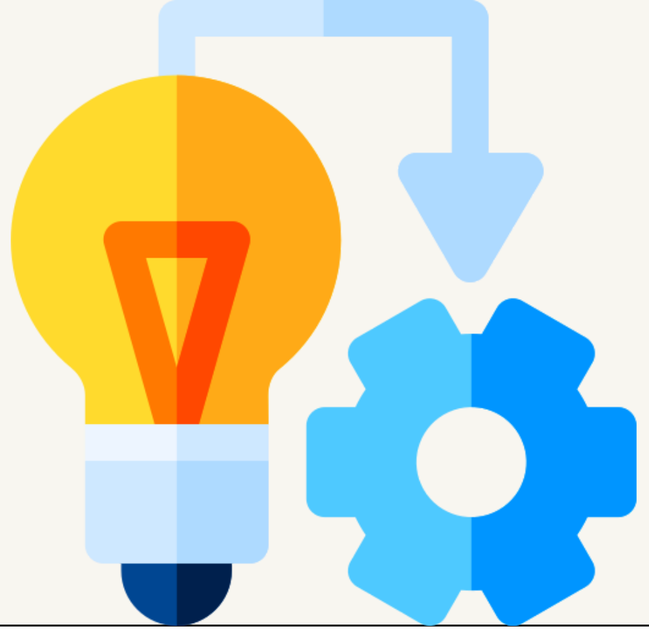
Linguagens e tecnologias utilizadas

Linguagens e estilização

- Linguagem da I.A : Python 
- Interface Web : HTML + CSS + JS 

Bibliotecas

Implementação



Primeira Etapa

- Escolha do DataSet a ser utilizado;
- Pré processamento dos dados para treinamento;
- Escolha do algoritmo de treinamento

Segunda Etapa

- Implementação da aplicação ;
- Treinamento da I.A com a base de teste;
- Teste com os demais algoritmos de treinamento;

Terceira Etapa

- Implementação da Interface de uso;
- Criação de Formulário na interface para inserção dos dados de entrada;

CÓDIGO FONTE

```python
def get_from_csv(data):
    """ Return predictors and targets from a single csv """
    predictors = data.iloc[:, 0:11].values
    targets = data.iloc[:, 11].values

    # escaler = preprocessing.MinMaxScaler()
    escaler = preprocessing.StandardScaler()
    escalerLabel = preprocessing.LabelBinarizer()

    age = escaler.fit_transform(np.array(predictors[:, 0]).reshape(-1, 1))
    escalerLabel.fit(['M', 'F'])
    sex = escalerLabel.transform(np.array(predictors[:, 1]).reshape(-1, 1))
    escalerLabel.fit(['ATA', 'NAP', 'ASY', 'TA'])
    chestPainType = escalerLabel.transform(np.array(predictors[:, 2]).reshape(-1, 1))
    restingBP = escaler.fit_transform(np.array(predictors[:, 3]).reshape(-1, 1))
    cholesterol = escaler.fit_transform(np.array(predictors[:, 4]).reshape(-1, 1))
    fastingBS = escaler.fit_transform(np.array(predictors[:, 5]).reshape(-1, 1))
    escalerLabel.fit(['Normal', 'ST', 'LVH'])
    restingECG = escalerLabel.transform(np.array(predictors[:, 6]).reshape(-1, 1))
    maxHR = escaler.fit_transform(np.array(predictors[:, 7]).reshape(-1, 1))
    escalerLabel.fit(['N', 'Y'])
    exerciseAngina = escalerLabel.fit_transform(np.array(predictors[:, 8]).reshape(-1, 1))
    oldpeak = escaler.fit_transform(np.array(predictors[:, 9]).reshape(-1, 1))
    escalerLabel.fit(['Up', 'Flat', 'Down'])
    sT_Slope = escalerLabel.transform(np.array(predictors[:, 10]).reshape(-1, 1))

    predictors = np.column_stack((age, sex, chestPainType, restingBP, cholesterol, fastingBS, restingECG, maxHR,
                                  exerciseAngina, oldpeak, sT_Slope))

    return predictors, targets
```

CÓDIGO FONTE

```python
def train_model(data_set_path, classifier):
    clf = MLPClassifier()
    data = pd.read_csv(data_set_path)

    # Splitting the dataset into  training and validation sets
    training_set, validation_set = train_test_split(data, test_size=0.2, random_state=21)

    X_train, Y_train = get_from_csv(training_set)
    X_val, Y_val = get_from_csv(validation_set)

    # Fitting the training data to the network
    classifier.fit(X_train, Y_train)

    # Predicting y for X_val
    Y_pred = classifier.predict(X_val)

    # Comparing the predictions against the actual observations in y_val
    cm = confusion_matrix(Y_pred, Y_val)

    # Printing the accuracy
    print(f"Accuracy of Classifier : {accuracy(cm)}")
    return classifier
```

CÓDIGO FONTE

```python
def accuracy(confusion_matrix):
    diagonal_sum = confusion_matrix.trace()
    sum_of_all_elements = confusion_matrix.sum()
    return diagonal_sum / sum_of_all_elements
```

CÓDIGO FONTE

```python
def MLP_training():
    # quantidade_de_neuronios_na_camada_oculta = (numero de entradas + numero de saida) / 2
    # Initializing the MLPClassifier
    classifier_mlp = MLPClassifier(hidden_layer_sizes=(10, 6, 3), max_iter=5000, activation='logistic', solver='adam',
                                   random_state=1, momentum=0.8, learning_rate_init=0.1, verbose=False)
    print('\n==MLP==')
    # save the model to disk
    pickle.dump(train_model("application/heart.csv", classifier_mlp), open('application/mlp_model.sav', 'wb'))
    result = evaluate(None)
    print(result)
    print('\n')
```

CÓDIGO FONTE

```python
def Perceotron_training():
    classifier_ppn = Perceptron(max_iter=5000, verbose=False, n_iter_no_change=10)
    print('\n==Perceptron==')
    train_model("../heart.csv", classifier_ppn)
    print('\n')
```

CÓDIGO FONTE

```python
def evaluate(json_path='../test.json', heart_csv_path='../heart.csv', mlp_model_path='../mlp_model.sav'):
    # load the model from disk
    classifier_mlp = pickle.load(open(mlp_model_path, 'rb'))
    # TODO: read json by API endpoint
    data = pd.read_json(json_path, orient='records')
    user_df = pd.json_normalize(data['data'])
    data = pd.read_csv(heart_csv_path)
    dfs = [data, user_df]
    df = pd.concat(dfs, ignore_index=True)
    x, _ = get_from_csv(df)

    # predict_proba return [probability for '0', probability for '1']
    result = classifier_mlp.predict_proba(x[918].reshape(1, -1))
    return f'A probabilidade de você ter algum problema no coração é de {"{:.2f}".format(result[0][1] * 100)}%.'
```

Fim da Apresentação

OBRIGADO ...