Relatório

Para treinamento do modelo de identificação de diagnósticos, inicialmente utilizaria o conhecimento em relação ao dado descrito na EDA realizada em cima dos dados. Iniciaria o vetorização do dado, utilizando a mesma pipeline de desenvolvida na análise exploratória.

```
X = vectorizer.fit_transform(corpus)
```

Após isso começaria a pipeline de desenvolvimento dos modelos de classificação, definiria um conjunto de modelos, tais como randomforest, extratress, lightgbm, xgboost. Todos os modelos seriam parametrizados de acordo com o

```
models_params ={'n_estimators':[10,20, 50, 100, None],
                'max_depth':[4, 6, 10, 15, 20, 50, None]}
```

```
GridSearchCV(model, X, y cv=5, n_jobs=-1)
clf_xgb = GridSearchCV(model, models_params, cv=5,n_jobs=-1, verbose = 1)
```

Assim teria o melhor cada um dos modelos com suas melhores parametrizações, eu utilizaria o Stacking para empilhar todos esses modelos de diversas formas para conseguir obter um modelo mais eficiente vindo da combinação dos outros do que cada um deles isoladamente. Como o exemplo a seguir:

```
from sklearn.ensemble import StackingClassifier
estimators = [
    ('extra', ExtraTreesClassifier(random_state=42, n_jobs=4)),
    ('lgb', lgb.LGBMClassifier(max_depth=50, n_estimators=50,
random_state=42, n_jobs=4)),
    ('xgb', xgboost.XGBClassifier(max_depth=5,
              n_estimators=50, n_jobs=4,, random_state=42))]

clf_stack = StackingClassifier(
    estimators=estimators,
final_estimator=RandomForestClassifier(random_state=42, n_jobs=4))
```

Essa seria a primeira abordagem, pois é menos custoso em questão de hardware para treinar do que algum modelo de deep learning e pode dar resultados eficientes. Caso os resultados fossem ineficientes e tivesse

hardware disponível para treinar, partiria para uma modelo de deep learning tais como um LSTM ou uma variação dos modelos BERT.