**Detailed description of the solution for the project**

**Goal:**

Predictive model creation based on the encounters and the information of patients hospitalized due to COVID 19

**Tasks:**

1. Find all patients that were admitted to the hospital because of Covid-19
   1. Hint: Use the encounters table only
   2. Filtering fields: REASONCODE = 840539006 and CODE = 1505002 or 305351004
2. Generate labels for each patient: 1 if the patient died because of Covid-19, 0 otherwise
3. Create a model to predict whether a patient died from Covid-19, given that they were admitted to the hospital because of Covid-19. Use only the following three covariates: gender, race, and the age at which the patient was admitted.

**Solution:** The solution contains two parts, model creation and model prediction GUI. The codes were first written in Jupyter Notebook style in colab, then formed into a package using Visual Studio 2019. The process includes the following steps

1. Imported the necessary modules/libraries
2. Read data from the two given files. Considering performance, only project relevant columns were read.
3. Found all patients that were admitted to the hospital because of Covid-19
4. Created a dataset by join the encounter and patient info
5. Labeled the patient by 1 if there is a death date, otherwise by 0, named the column as DIED
6. Calculated the age based on the date difference of the START date from the encounter info and the birth date of the patient
7. Reduced the dimensions, the dataset only contains three covariates: named GENDER, RACE and AGE, plus the label DIED
8. Checked the features of the dataset using seaborn and matplotlib to visualize the data patterns, one important pattern is about the average age
   1. 48 for survived patients
   2. 73 for patients died because of COVID19
9. Prepared a dataset for data modeling
   1. Used One Hot Encoder to transform the categorical RACE column into 4 columns of numbers
   2. Encoded GENDER into one column of values (F: 0, M:1)
   3. Split dataset into two subsets: X\_train and X\_test, with rows of 75% and 25% respectively. Rows are selected randomly using scikit-learn train\_selection\_split module
10. 4 classical Machine Learning algorithms were used to build a model for each. Confusion matrix and accuracy score were calculated for each of the model created.
    1. Model 1: Logistic Regression
       1. LogisticRegression module from scikit-learn was used to create a model, accuracy score is 0.778 or 77.8%
       2. Applied K-fold cross validation: Accuracy: 78.97 %, Standard Deviation: 1.57 %
       3. To explore the important features, standard scaling was performed. The coefficients show that the AGE is an important feature, next most important feature is GENDER
       4. Reduced dimension by excluding the 4 columns for RACE, repeated the modeling process. Confusion matrix and accuracy score doesn’t change much. So, in the rest of modeling process, RACE was kept
    2. Model 2: Decision Tree (from sklearn.tree import DecisionTreeClassifier)

Repeated the same process, the accuracy score is 0.79, or 79%

* 1. Model 3: K-Nearest Neighbors (from sklearn.neighbors import KNeighborsClassifier). For this model, the hyperparameter, number of K Neighbors was tested, the best value is 38 with model accuracy score 0.783, 0r 78.3%
  2. Model 4: Support Vector Machine (from sklearn.svm import SVC), accuracy: 0.778 or 77.8%

All 4 classical classifiers give model accuracy in [77.8%, 79%]

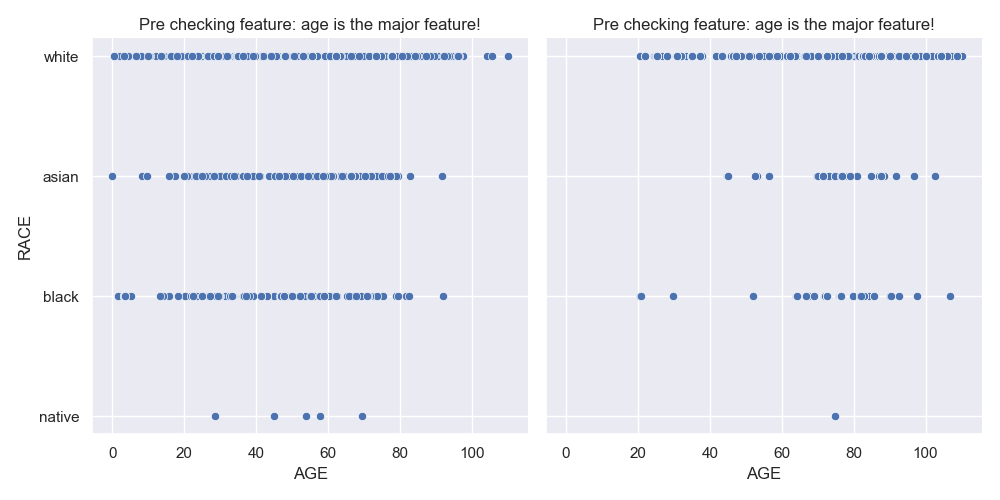
1. Because of curiosity, I tried the Deep Learning modeling. Variated the number of hidden network layers, and/or the number of nodes in a layer, the epoch and batch size. Achieved accuracy 79.3%, better than those of classic classifiers, but not significant
2. All models created can be saved using pickle

**Production:**

1. The Jupyter Notebook file was included, named create\_model.ipynb, which can be opened and run with the two original files in the same folder.
2. A whole package that contains
   1. The model creation (create.py), which does the similar process as the codes do in the create\_model.ipynb file
   2. GUI, an application (predict.py), provides preliminary graphic user interface, so that the user can select the gender and race, and enter the age, then click the Predict button to see the results from 4 classic models (Survive or Die)
3. The whole package is pushed to GitHub at <https://github.com/karlwang1xx1/covid19/tree/master/covid19>

It is in private mode, but access can be granted to individual(s) by invite. The package is version controlled and can be cloned conveniently.

**Appendices:** Figures will be created in the process of building models

Here are a few:  Chart, histogram

Description automatically generated

Screenshot of the UI

Graphical user interface, application, Word

Description automatically generated