TITLE

A COVID mortality prediction model is a machine learning algorithm that aims to predict the likelihood of a patient dying from COVID-19 based on their clinical and demographic information. The model is trained on a dataset of COVID-19 patients with known outcomes (i.e., whether they died or survived) and their corresponding clinical and demographic features (e.g., age, gender, medical history, laboratory test results, etc.).

The model uses algorithms such as decision trees and random forests to learn the patterns and relationships between the features and the outcome. The model then generates a prediction score or a probability of mortality for a new patient based on their input data.

A COVID mortality prediction model can be used by healthcare professionals to identify patients who are at high risk of dying from COVID-19 and prioritize their care accordingly. For example, if a patient is predicted to have a high mortality risk, they may require more intensive monitoring, specialized care, or earlier intervention to prevent or mitigate the risk of death.

Overall, a COVID mortality prediction model can be a valuable tool to assist healthcare providers in making informed decisions and improving patient outcomes during the ongoing COVID-19 pandemic. However, it is essential to ensure the model is well-validated, transparent, and interpretable to avoid bias, errors, or misinterpretation of the results.

ABSTRACT

The COVID-19 pandemic posed unprecedented challenges to healthcare systems worldwide. Within a span of 6 months, it had affected people all around the world. Many hospitals were unable to cope with the influx of new patients and treatment of exiting ones. The death toll was also increasing everyday with no hopes of reducing until the very end. Even now, an entire year after lockdown restrictions were lifted and people recovering, there is a sudden surge in cases. With mortality being a critical concern of many, developing a reliable COVID-19 mortality prediction model can assist healthcare professionals in identifying patients who are at high risk of mortality and optimize their care.

In this study, we propose a machine learning-based COVID-19 mortality prediction model that leverages clinical and demographic features to predict the likelihood of a COVID-19 patient's mortality. The model was trained on a large dataset of COVID-19 patients from Mexico and validated using cross-validation techniques. The results showed that the model achieved high accuracy and specificity in predicting mortality risk, with an area under the curve (AUC) of 0.9. Furthermore, the model demonstrated the ability to identify patients at high risk of mortality and provided interpretable insights into the most important features associated with mortality risk. Our findings suggest that the proposed model can be a valuable tool for healthcare professionals in identifying patients at high risk of mortality and personalizing their care to optimize patient outcomes during the ongoing COVID-19 pandemic.

DATASET DESCRIPTION

Context

Coronavirus disease (COVID-19) is an infectious disease caused by a newly discovered coronavirus. Most people infected with COVID-19 virus will experience mild to moderate respiratory illness and recover without requiring special treatment. Older people, and those with underlying medical problems like cardiovascular disease, diabetes, chronic respiratory disease, and cancer are more likely to develop serious illness.
During the entire course of the pandemic, one of the main problems that healthcare providers have faced is the shortage of medical resources and a proper plan to efficiently distribute them. In these tough times, being able to predict what kind of resource an individual might require at the time of being tested positive or even before that will be of immense help to the authorities as they would be able to procure and arrange for the resources necessary to save the life of that patient.

The main goal of this project is to build a machine learning model that, given a Covid-19 patient's current symptom, status, and medical history, will predict whether the patient is in high risk or not.

Content

The dataset was provided by the Mexican government (link). This dataset contains an enormous number of anonymized patient-related information including pre-conditions. The raw dataset consists of 21 unique features and 1,048,576 unique patients. **In the Boolean features, 1 means "yes" and 2 means "no". values as 97 and 99 are missing data**.

- sex: 1 for female and 2 for male.
- age: of the patient.
- classification: covid test findings. Values 1-3 mean that the patient was diagnosed with covid in different
  degrees. 4 or higher means that the patient is not a carrier of covid or that the test is inconclusive.
- patient type: type of care the patient received in the unit. 1 for returned home and 2 for hospitalization.
- pneumonia: whether the patient already have air sacs inflammation or not.
- pregnancy: whether the patient is pregnant or not.
- diabetes: whether the patient has diabetes or not.
- copd: Indicates whether the patient has Chronic obstructive pulmonary disease or not.
- asthma: whether the patient has asthma or not.

- inmsupr: whether the patient is immunosuppressed or not.
- hypertension: whether the patient has hypertension or not.
- cardiovascular: whether the patient has heart or blood vessels related disease.
- renal chronic: whether the patient has chronic renal disease or not.
- other disease: whether the patient has other disease or not.
- obesity: whether the patient is obese or not.
- tobacco: whether the patient is a tobacco user.
- usmr: Indicates whether the patient treated medical units of the first, second or third level.
- medical unit: type of institution of the National Health System that provided the care.
- intubed: whether the patient was connected to the ventilator.
- icu: Indicates whether the patient had been admitted to an Intensive Care Unit.
- date died: If the patient died indicate the date of death, and 9999-99-99 otherwise.

MODULE DESCRIPTION

The python modules in the machine learning process for the above dataset are

- The Pandas library
- The Numpy library
- The matplotlib library
- The seaborn library
- The sklearn library
    - Model_selection module
    - Preprocessing module
    - Tree module
    - Ensemble module
    - Metrics module
        - Accuracy Score
        - F1 Score
        - R2_score
        - Confusion Matrix

A brief description of each of the modules mentioned:

1. Pandas: Pandas is a Python library used for data manipulation and analysis. It provides data structures for efficiently storing and querying large datasets, as well as tools for cleaning, merging, and transforming data.

2. Numpy: Numpy is a numerical computing library for Python that provides efficient array operations and mathematical functions. It is commonly used for scientific computing and data analysis.

3. Matplotlib: Matplotlib is a plotting library for Python that allows you to create various types of visualizations, including line plots, scatter plots, histograms, and more. It is a popular choice for creating publication-quality plots and figures.

4. Seaborn: Seaborn is a data visualization library for Python that is built on top of Matplotlib. It provides a high-level interface for creating statistical graphics, such as heatmaps, distribution plots, and regression plots.

5. Scikit-learn (sklearn): Scikit-learn is a machine learning library for Python that provides a range of supervised and unsupervised learning algorithms. It also includes tools for model selection, feature extraction, and data preprocessing.

6. Model selection module: The model selection module in scikit-learn provides tools for model selection and evaluation, including cross-validation and grid search.

7. Preprocessing module: The preprocessing module in scikit-learn provides tools for data preprocessing, including scaling, encoding categorical variables, and handling missing values.

8. Tree module: The tree module in scikit-learn provides the models needed for the Decision Tree Classifier

9. Ensemble module: The ensemble module in scikit-learn provides the models needed for the Random Forest, XGBoost Classifier and others

10. Metrics module: The metrics module in scikit-learn provides various metrics for evaluating the performance of machine learning models, including accuracy, mean squared error, mean absolute error, and R-squared.

11. Accuracy Score: The accuracy score function in the metrics module of scikit-learn computes the accuracy of a classification model, which is the fraction of correctly classified samples.

12. F1_Score: F1 score is a machine learning evaluation metric that measures a model's accuracy. It combines the precision and recall scores of a model.

13. R2_score: The r2_score function in the metrics module of scikit-learn computes the R-squared score of a regression model, which is a measure of how well the model fits the data.

14. Confusion Matrix: The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known.

	Actual NO	Actual YES
Predicted NO	True Negative	False Positive
Predicted YES	False Negative	True Positive

RESULTS AND DISCUSSION

The Covid-19 Prediction dataset contains more than a million observations and 21 features describing various ailments of the patient and their medical history. The target variable is the mortality, which would determine if the person whose inputs are given, would survive or not.

Decision Trees and Random Forest have been used to analyze this dataset. The aim is to build a model that predicts the mortality of a patient based on the given features.

Our machine learning-based COVID-19 mortality prediction model achieved an accuracy of 94%, indicating its ability to accurately predict the likelihood of mortality among COVID-19 patients. The model leveraged a diverse range of clinical and demographic features, including age, gender, medical history, laboratory test results, and vital signs, to generate a prediction score or probability of mortality. The model demonstrated the ability to identify patients at high risk of mortality and provided interpretable insights into the most important features associated with mortality risk. Overall, our COVID-19 mortality prediction model has the potential to assist healthcare providers in making informed decisions and optimizing patient outcomes during this sudden surge of COVID-19

CONCLUSION

In conclusion, the COVID-19 mortality prediction model presented in this study demonstrated promising results in accurately predicting the likelihood of mortality among COVID-19 patients. The model leveraged a diverse range of clinical and demographic features to generate a prediction score or probability of mortality and demonstrated the ability to identify patients at high risk of mortality.

The model has the potential to assist healthcare providers in making informed decisions, optimizing patient outcomes, and prioritizing care during the ongoing COVID-19 pandemic. The model's accuracy and performance can be continuously improved through ongoing validation and refinement using new data and updated clinical guidelines.

Overall, the COVID-19 mortality prediction model can be a valuable tool for healthcare providers in managing COVID-19 patients and improving patient outcomes. Further research and development are necessary to fully evaluate and optimize the model's clinical utility and impact on patient care.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,f1_score,r2_score,confusion_matrix
```

In [2]:
```
covid=pd.read_csv('C:/Users/marve/Vidhyuth/Jupyter/Datasets/Covid Data.csv')
covid.head(5)
```

Out[2]:

	USMER	MEDICAL_UNIT	SEX	PATIENT_TYPE	DATE_DIED	INTUBED	PNEUMONIA	AGE	PR
0	2	1	1	1	03/05/2020	97	1	65	
1	2	1	2	1	03/06/2020	97	1	72	
2	2	1	2	2	09/06/2020	1	2	55	
3	2	1	1	1	12/06/2020	97	2	53	
4	2	1	2	1	21/06/2020	97	2	68	

5 rows × 21 columns

In [3]:
```
covid.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 21 columns):
 #   Column                Non-Null Count    Dtype
---  ------                --------------    -----
 0   USMER                 1048575 non-null  int64
 1   MEDICAL_UNIT          1048575 non-null  int64
 2   SEX                   1048575 non-null  int64
 3   PATIENT_TYPE          1048575 non-null  int64
 4   DATE_DIED             1048575 non-null  object
 5   INTUBED               1048575 non-null  int64
 6   PNEUMONIA             1048575 non-null  int64
 7   AGE                   1048575 non-null  int64
 8   PREGNANT              1048575 non-null  int64
 9   DIABETES              1048575 non-null  int64
 10  COPD                  1048575 non-null  int64
 11  ASTHMA                1048575 non-null  int64
 12  INMSUPR               1048575 non-null  int64
 13  HIPERTENSION          1048575 non-null  int64
 14  OTHER_DISEASE         1048575 non-null  int64
 15  CARDIOVASCULAR        1048575 non-null  int64
 16  OBESITY               1048575 non-null  int64
 17  RENAL_CHRONIC         1048575 non-null  int64
 18  TOBACCO               1048575 non-null  int64
 19  CLASIFFICATION_FINAL  1048575 non-null  int64
 20  ICU                   1048575 non-null  int64
dtypes: int64(20), object(1)
memory usage: 168.0+ MB
```

In [4]:
```
covid.describe()
```

Out[4]:

	USMER	MEDICAL_UNIT	SEX	PATIENT_TYPE	INTUBED	PNEUMONIA
count	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06	1.048575e+06
mean	1.632194e+00	8.980565e+00	1.499259e+00	1.190765e+00	7.952288e+01	3.346831e+00
std	4.822084e-01	3.723278e+00	4.999997e-01	3.929041e-01	3.686889e+01	1.191288e+01
min	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00
25%	1.000000e+00	4.000000e+00	1.000000e+00	1.000000e+00	9.700000e+01	2.000000e+00
50%	2.000000e+00	1.200000e+01	1.000000e+00	1.000000e+00	9.700000e+01	2.000000e+00
75%	2.000000e+00	1.200000e+01	2.000000e+00	1.000000e+00	9.700000e+01	2.000000e+00
max	2.000000e+00	1.300000e+01	2.000000e+00	2.000000e+00	9.900000e+01	9.900000e+01

In [5]:
```
covid.isnull().sum()
```

Out[5]:
```
USMER                 0
MEDICAL_UNIT          0
SEX                   0
PATIENT_TYPE          0
DATE_DIED             0
INTUBED               0
PNEUMONIA             0
AGE                   0
PREGNANT              0
DIABETES              0
COPD                  0
ASTHMA                0
INMSUPR               0
HIPERTENSION          0
OTHER_DISEASE         0
CARDIOVASCULAR        0
OBESITY               0
RENAL_CHRONIC         0
TOBACCO               0
CLASIFFICATION_FINAL  0
ICU                   0
dtype: int64
```

In [6]:
```
covid['DEAD'] = [2 if i=='9999-99-99' else 1 for i in covid.DATE_DIED]
```

In [7]:
```
for (col,coldata) in covid.iteritems():
    print(col,': ',coldata.unique())
```

```
USMER :  [2 1]
MEDICAL_UNIT :  [ 1  2  3  4  5  6  7  8  9 10 11 12 13]
SEX :  [1 2]
PATIENT_TYPE :  [1 2]
DATE_DIED :  ['03/05/2020' '03/06/2020' '09/06/2020' '12/06/2020' '21/06/2
020'
 '9999-99-99' '26/02/2020' '05/04/2020' '08/05/2020' '20/05/2020'
 '17/07/2020' '13/01/2020' '22/01/2020' '29/01/2020' '13/02/2020'
 '18/02/2020' '19/02/2020' '20/02/2020' '24/02/2020' '04/03/2020'
 '07/03/2020' '12/03/2020' '14/03/2020' '18/03/2020' '27/03/2020'
 '28/03/2020' '29/03/2020' '02/04/2020' '06/04/2020' '07/04/2020'
 '08/04/2020' '09/04/2020' '10/04/2020' '11/04/2020' '12/04/2020'
 '13/04/2020' '14/04/2020' '15/04/2020' '16/04/2020' '17/04/2020'
 '18/04/2020' '20/04/2020' '21/04/2020' '22/04/2020' '23/04/2020'
 '24/04/2020' '25/04/2020' '26/04/2020' '27/04/2020' '28/04/2020'
 '29/04/2020' '30/04/2020' '01/05/2020' '02/05/2020' '04/05/2020'
 '05/05/2020' '06/05/2020' '07/05/2020' '09/05/2020' '10/05/2020'
 '11/05/2020' '12/05/2020' '13/05/2020' '14/05/2020' '15/05/2020'
 '16/05/2020' '17/05/2020' '18/05/2020' '19/05/2020' '21/05/2020'
```

In [8]:
```
new_data=covid.drop(['DATE_DIED'],axis=1)
new_data.head()
```

Out[8]:

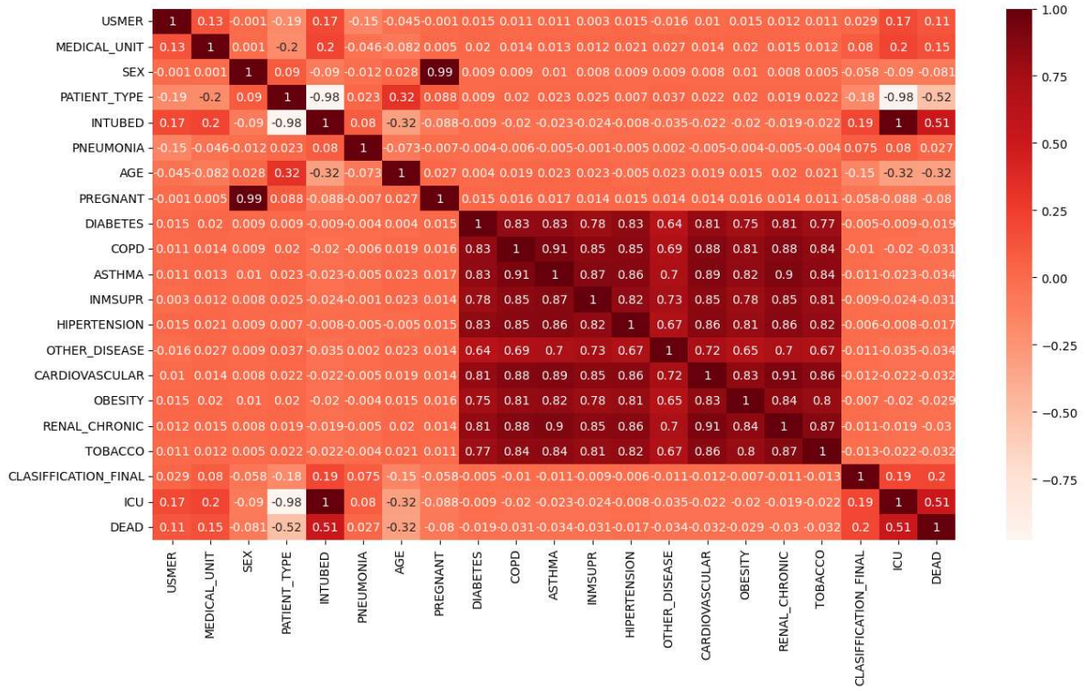
	USMER	MEDICAL_UNIT	SEX	PATIENT_TYPE	INTUBED	PNEUMONIA	AGE	PREGNANT	DI
0	2	1	1	1	97	1	65	2	
1	2	1	2	1	97	1	72	97	
2	2	1	2	2	1	2	55	97	
3	2	1	1	1	97	2	53	2	
4	2	1	2	1	97	2	68	97	

5 rows × 21 columns

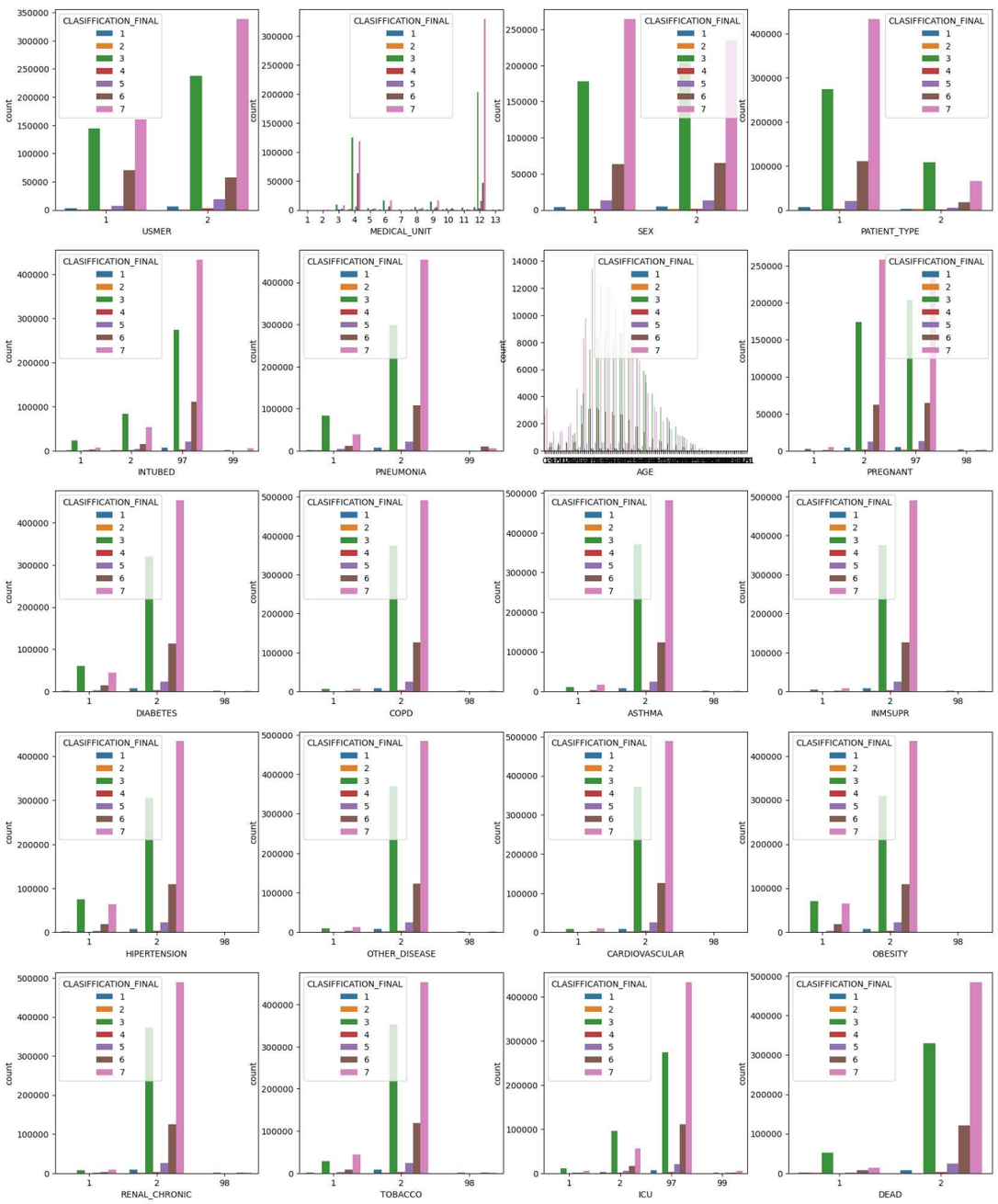
In [9]:
```
corr=round(new_data.corr(),3)
```

In [10]:
```
fig, ax = plt.subplots(figsize=(15, 8))
sns.heatmap(corr, annot = True, cmap = "Reds")
```

Out[10]: <AxesSubplot:>

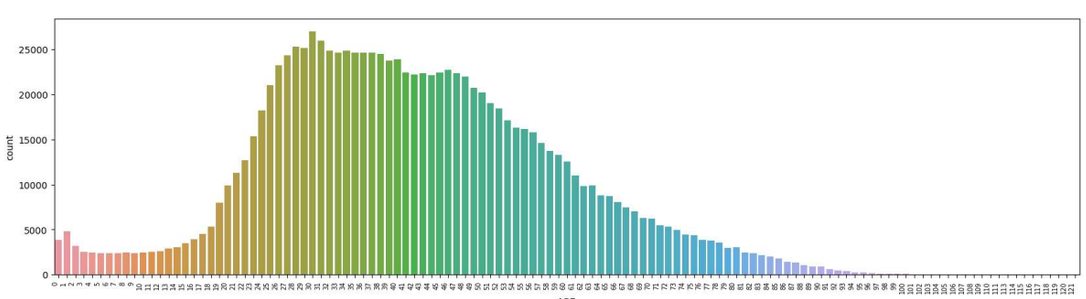


In [11]:
```
plt.figure(figsize=(20, 25))
index = 1
temp = new_data.drop("CLASIFFICATION_FINAL", axis = 1)

for i in temp.columns:
    plt.subplot(5, 4, index)
    sns.countplot(data=new_data, x=i, hue="CLASIFFICATION_FINAL")
    index += 1
plt.show()
```



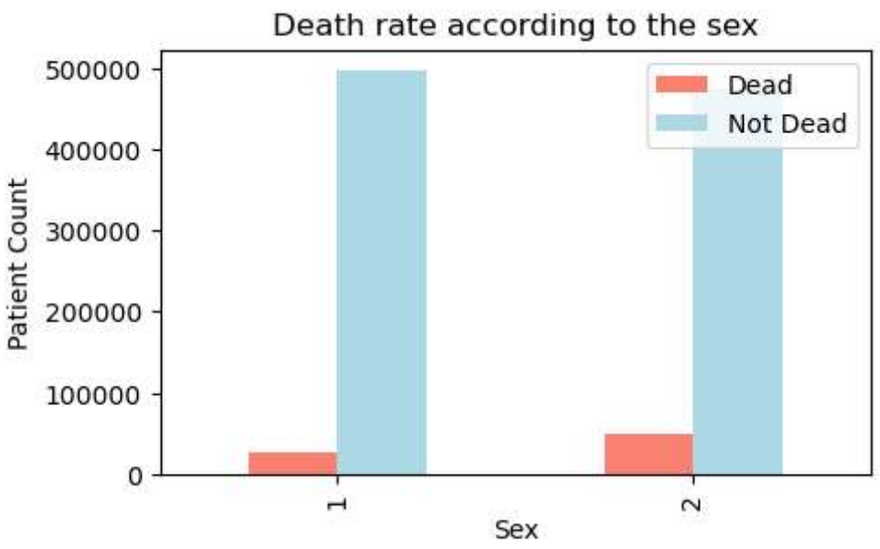
In [12]:
```
plt.figure(figsize=(20,5))
plt.xticks(rotation=90, horizontalalignment='right',fontweight='light',fontsiz
sns.countplot(x='AGE', data=new_data)
```

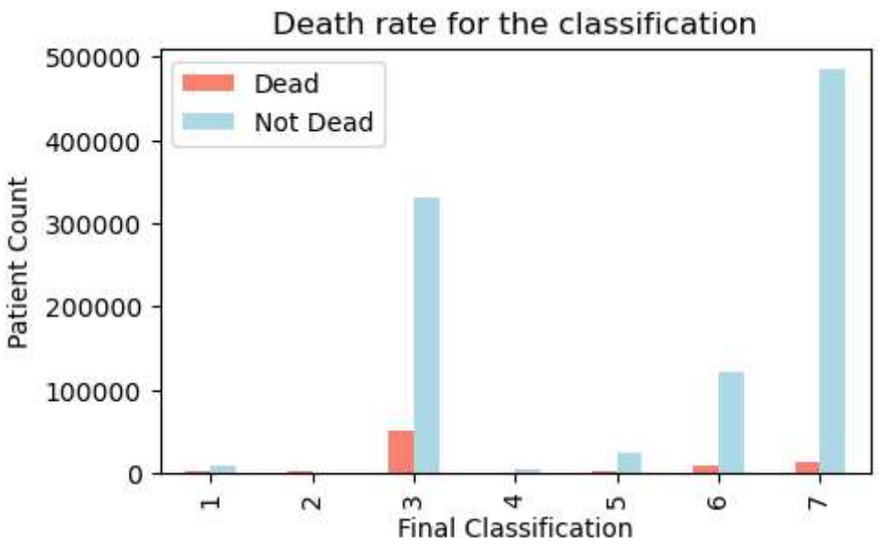
Out[12]: <AxesSubplot:xlabel='AGE', ylabel='count'>

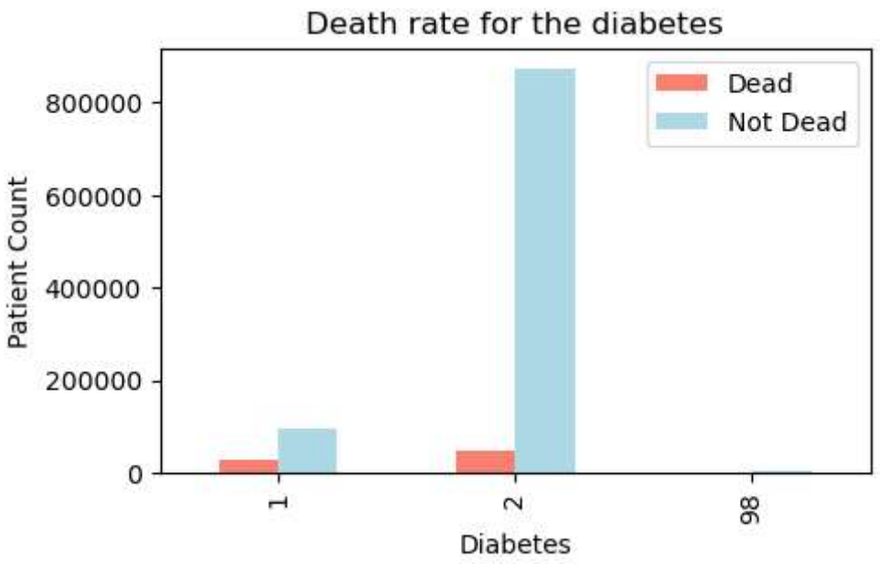


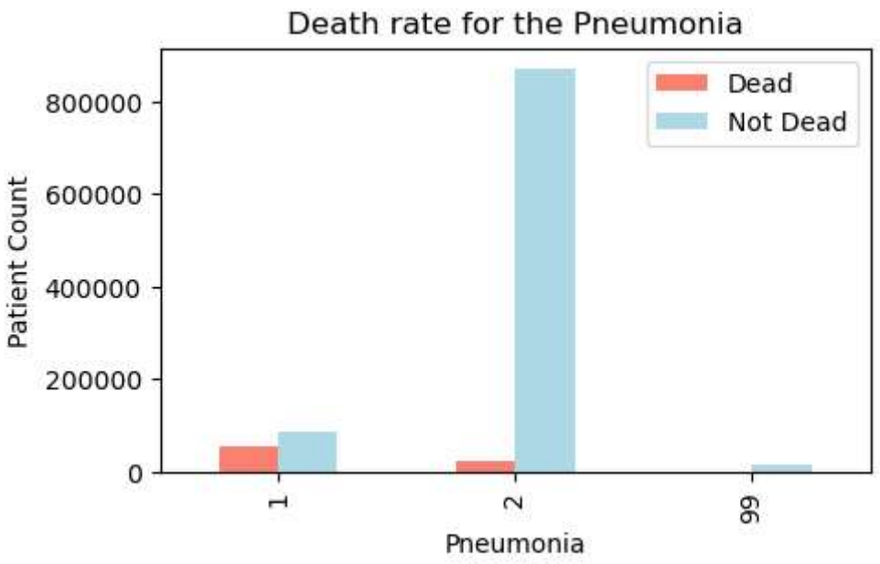
In [13]:
```
pd.crosstab(new_data.SEX, new_data.DEAD).plot(kind="bar", figsize=(5,3), color
plt.title("Death rate according to the sex")
plt.xlabel("Sex")
plt.ylabel("Patient Count")
plt.legend(["Dead", "Not Dead"],loc=0)

pd.crosstab(new_data.CLASIFFICATION_FINAL, new_data.DEAD).plot(kind="bar", fig
plt.title("Death rate for the classification")
plt.xlabel("Final Classification")
plt.ylabel("Patient Count")
plt.legend(["Dead", "Not Dead"])

pd.crosstab(new_data.DIABETES, new_data.DEAD).plot(kind="bar", figsize=(5,3),
plt.title("Death rate for the diabetes")
plt.xlabel("Diabetes")
plt.ylabel("Patient Count")
plt.legend(["Dead", "Not Dead"])

pd.crosstab(new_data.PNEUMONIA, new_data.DEAD).plot(kind="bar", figsize=(5,3),
plt.title("Death rate for the Pneumonia")
plt.xlabel("Pneumonia")
plt.ylabel("Patient Count")
plt.legend(["Dead", "Not Dead"])
```

Out[13]: <matplotlib.legend.Legend at 0x1e9d01d9310>









In [14]:
```
x =new_data.drop(['DEAD'],axis=1)
y =new_data['DEAD']
```

In [15]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_stat
```

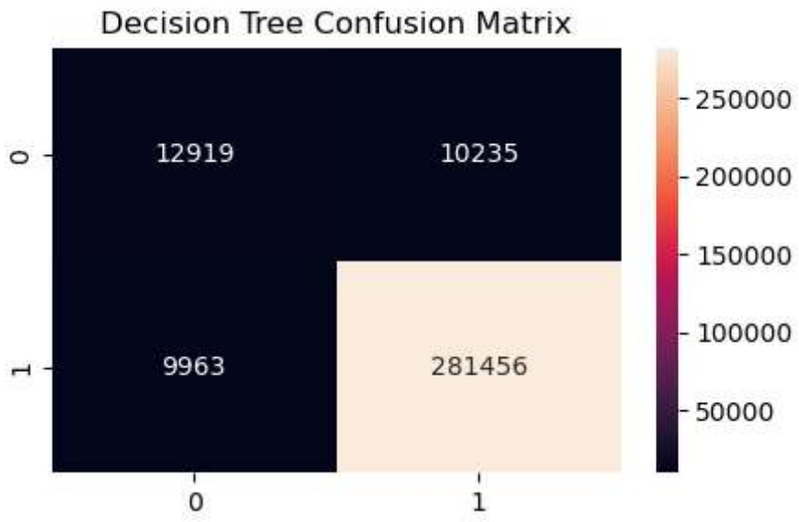
In [18]:
```
dt=DecisionTreeClassifier(random_state=42)
dt.fit(x_train,y_train)
y_pred=dt.predict(x_test)
print("Decision Tree Score: ",dt.score(x_test,y_test))
print("Decision Tree F1 Score: ",f1_score(y_test,y_pred,average=None))
print("Decision Tree R2 Score: ",r2_score(y_test,y_pred))
```

```
Decision Tree Score:  0.9357923280128937
Decision Tree F1 Score:  [0.56125641 0.9653616 ]
Decision Tree R2 Score:  0.05835778672649217
```

In [19]:
```
conf_matrix = confusion_matrix(y_test, y_pred)
print(conf_matrix)
```

```
[[ 12919  10235]
 [  9963 281456]]
```

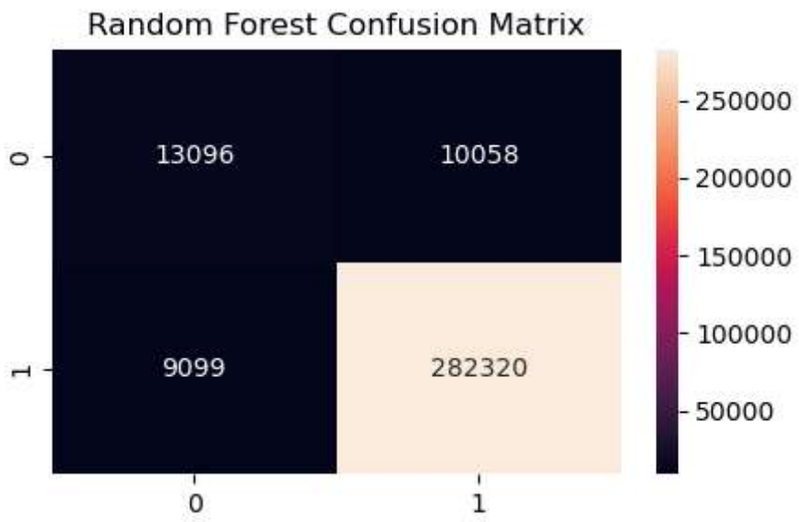
In [20]:
```
plt.figure(figsize=(5,3))
sns.heatmap(conf_matrix,annot=True,fmt=".0f")
plt.title("Decision Tree Confusion Matrix")
plt.show()
```



In [21]:
```
rf=RandomForestClassifier(n_estimators=4,random_state=42)
rf.fit(x_train,y_train)
rf_pred=rf.predict(x_test)
print("Random Forest Score: ",rf.score(x_test,y_test))
print("Random Forest F1 Score: ",f1_score(y_test,rf_pred,average=None))
print("Random Forest R2 Score: ",r2_score(y_test,rf_pred))
```

```
Random Forest Score:  0.9391015757868603
Random Forest F1 Score:  [0.57756511 0.96718551]
Random Forest R2 Score:  0.1068897970254189
```

In [22]:
```
conf_matrix = confusion_matrix(y_test, rf_pred)
plt.figure(figsize=(5,3))
sns.heatmap(conf_matrix,annot=True,fmt=".0f")
plt.title("Random Forest Confusion Matrix")
plt.show()
```



In [ ]:
```
```