In [96]:
```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

import pandas as pd
import numpy as np
```

In [97]:
```
df= pd.read_csv("./diabetic_data.csv")
df.head(2)
```

Out[97]:

	encounter_id	patient_nbr	race	gender	age	weight	admission_type_id	discharge_disposition_i
0	2278392	8222157	Caucasian	Female	[0-10)	?	6	2
1	149190	55629189	Caucasian	Female	[10-20)	?	1	

2 rows × 50 columns

In [98]:
```
df.shape
```

Out[98]:
```
(101766, 50)
```

In [99]:
```
df.columns
```

Out[99]:
```
Index(['encounter_id', 'patient_nbr', 'race', 'gender', 'age', 'weight',
       'admission_type_id', 'discharge_disposition_id', 'admission_source_id',
       'time_in_hospital', 'payer_code', 'medical_specialty',
       'num_lab_procedures', 'num_procedures', 'num_medications',
       'number_outpatient', 'number_emergency', 'number_inpatient', 'diag_1',
       'diag_2', 'diag_3', 'number_diagnoses', 'max_glu_serum', 'A1Cresult',
       'metformin', 'repaglinide', 'nateglinide', 'chlorpropamide',
       'glimepiride', 'acetohexamide', 'glipizide', 'glyburide', 'tolbutamide',
       'pioglitazone', 'rosiglitazone', 'acarbose', 'miglitol', 'troglitazone',
       'tolazamide', 'examide', 'citoglipton', 'insulin',
       'glyburide-metformin', 'glipizide-metformin',
       'glimepiride-pioglitazone', 'metformin-rosiglitazone',
       'metformin-pioglitazone', 'change', 'diabetesMed', 'readmitted'],
      dtype='object')
```

In [100…
```
df=df.replace("?",np.nan)
```

In [101…
```
#dropping columns with large number of missing values
df = df.drop(['weight','payer_code','medical_specialty'], axis = 1)
```

In [102…
```
#variables (drugs named citoglipton and examide), all records have the same value
df = df.drop(['citoglipton', 'examide'], axis = 1)
```

In [103…
```
#Analyzing race column
df['race'].isnull().sum()
df['race'].value_counts()

df['race']=df['race'].fillna('UNK') #filling null with unk
```

In [104…
```
df['readmitted'].value_counts()
```

Out[104…
```
NO     54864
>30    35545
<30    11357
Name: readmitted, dtype: int64
```

In [105…
```
#Feature engineering
#Generating output variable
#we need to check whether a patient admitted within 30 days or not
df['target']=(df['readmitted']=='<30').astype('int')

#dropping readmitted column
df.drop(['readmitted'],axis=1,inplace=True)
df['readmitted']=df['target']
df.drop(['target'],axis=1,inplace=True)
```

In [106…
```
df['readmitted'].value_counts()
```

Out[106…
```
0    90409
1    11357
Name: readmitted, dtype: int64
```

In [107…
```
cleanup_age = {"age":     {"[0-10)": 5, "[10-20)": 15,"[20-30)": 25,"[30-40)": 35,"[40-
    "[60-70)": 65,"[70-80)": 75,"[80-90)": 85,"[90-100)": 95}}

df.replace(cleanup_age, inplace=True)
```

In [108…
```
#analyzing gender column
df['gender'].value_counts()
#removing invalid/unknown entries for gender
df=df[df['gender']!='Unknown/Invalid']
```

In [109…
```
df.head()
```

Out[109…

	encounter_id	patient_nbr	race	gender	age	admission_type_id	discharge_disposition_id
0	2278392	8222157	Caucasian	Female	5	6	25
1	149190	55629189	Caucasian	Female	15	1	1
2	64410	86047875	AfricanAmerican	Female	25	1	1

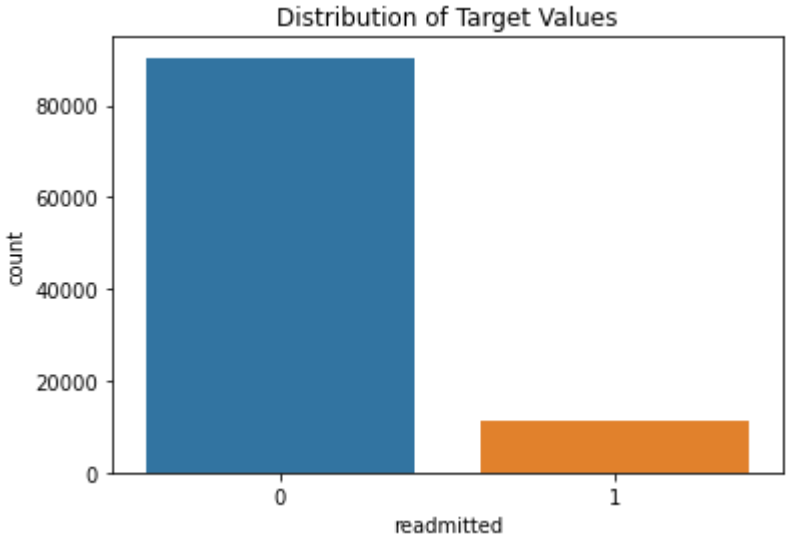
	encounter_id	patient_nbr	race	gender	age	admission_type_id	discharge_disposition_id
3	500364	82442376	Caucasian	Male	35	1	1
4	16680	42519267	Caucasian	Male	45	1	1

5 rows × 45 columns

Exploratory Data Analysis

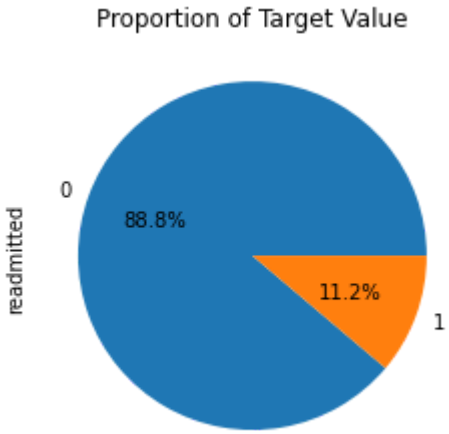
In [110…

```
sns.countplot(x = "readmitted", data = df)
plt.title("Distribution of Target Values")
plt.show()
```



In [111…

```
# Pie chart
df.readmitted.value_counts().plot.pie(autopct = "%.1f%%")
plt.title("Proportion of Target Value")
plt.show()
```



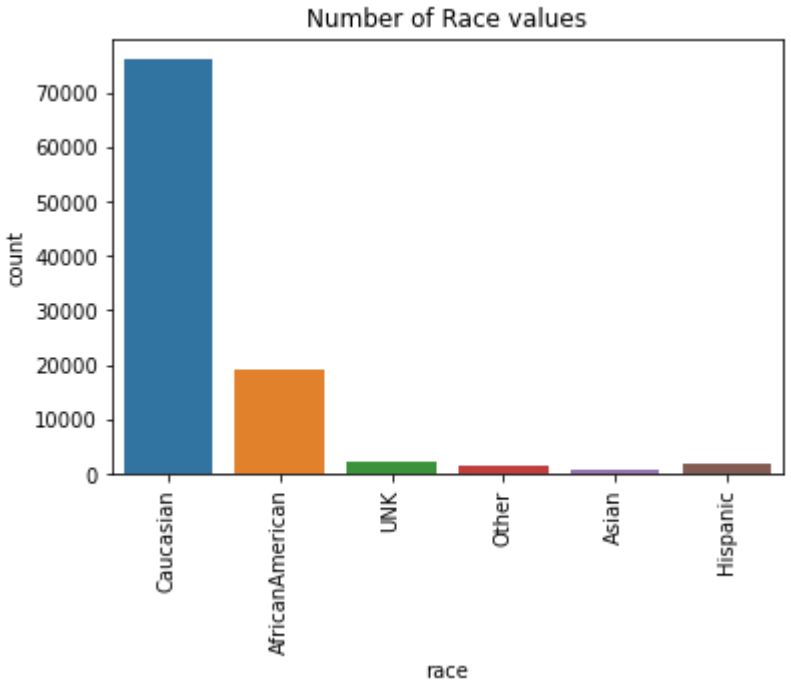
Our target variable is almost balanced distributed.

Race We have 5 different races value, these are;

Caucasian AfricanAmerican Hispanic Asian Other

In [112…

```
sns.countplot(x=df.race, data = df)
plt.xticks(rotation=90)
plt.title("Number of Race values")
plt.show()

print("Proportion of Race")
print(df.race.value_counts(normalize = True)*100)
```

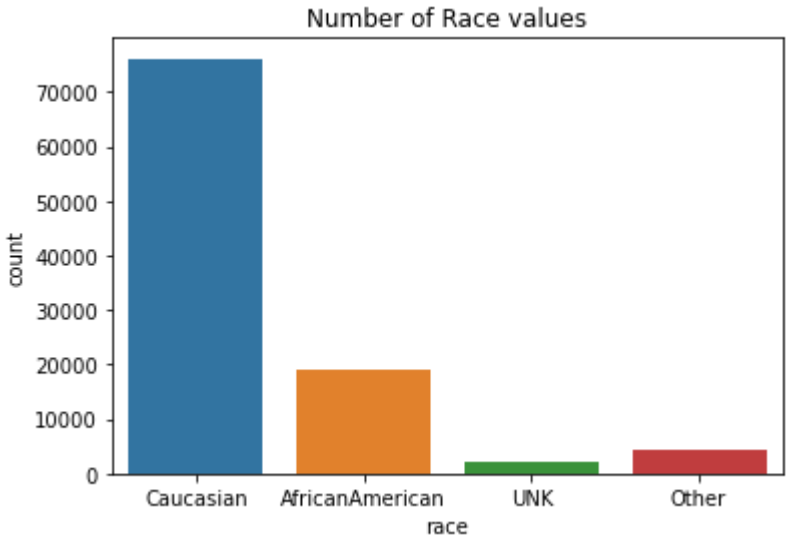



```
Proportion of Race
Caucasian          74.780618
AfricanAmerican    18.877195
UNK                 2.231656
Hispanic            2.001710
Other               1.478927
Asian               0.629895
Name: race, dtype: float64
```

As we see, there is Caucasians in 76 percent of all our data. And other other 24 percent is divided into African Americans, Hispanics, Asians and Others. Here we decided to divide into 3 groups like Caucasian, African American and Other.

In [113…

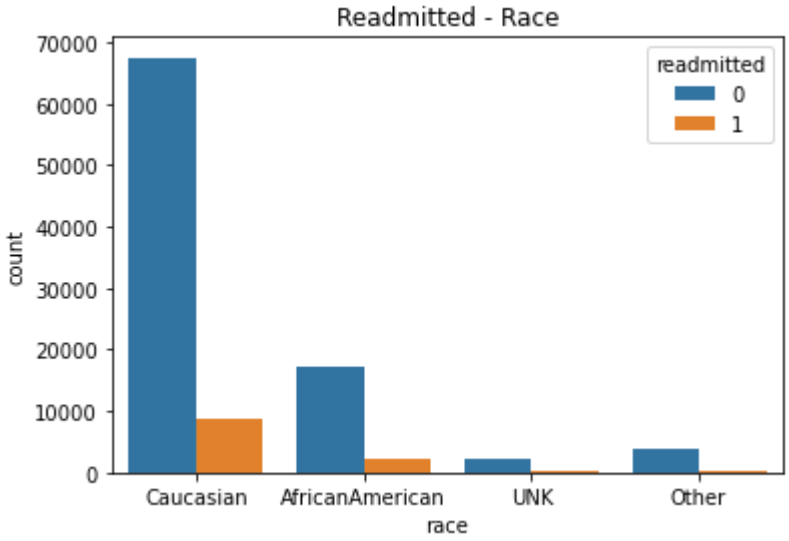
```
mapped_race = {"Asian":"Other","Hispanic":"Other"}
df.race = df.race.replace(mapped_race)

sns.countplot(x="race", data = df)
plt.title("Number of Race values")
plt.show()

print("Proportion of Race After the Mapping")
print(df.race.value_counts(normalize= True)*100)
```




```
Proportion of Race After the Mapping
Caucasian          74.780618
AfricanAmerican    18.877195
Other               4.110531
UNK                 2.231656
Name: race, dtype: float64
```

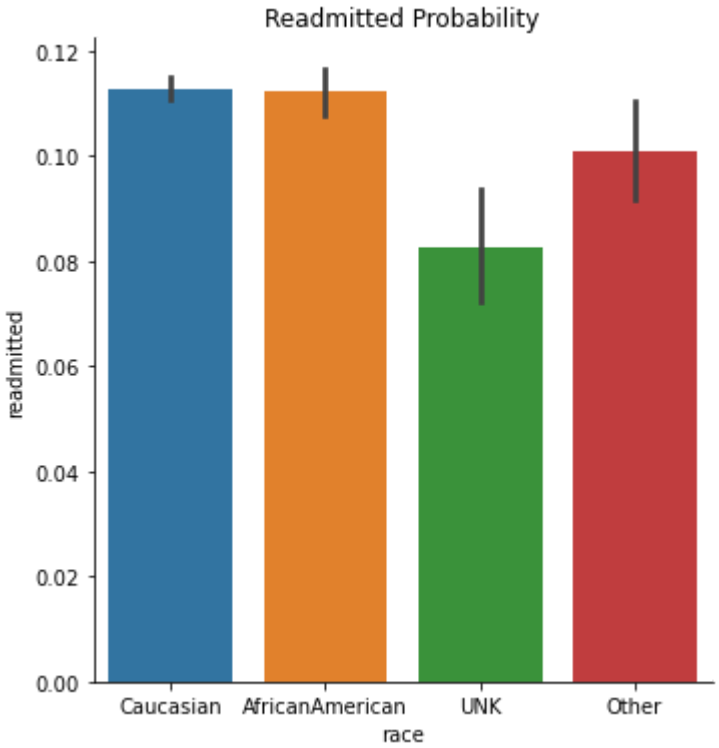
In [114…

```
sns.countplot(x="race", hue= "readmitted", data = df)
plt.title("Readmitted - Race")
plt.show()
```



In [115…

```
sns.catplot(x = "race", y = "readmitted",data = df, kind = "bar", height= 5)
plt.title("Readmitted Probability")
plt.show()
```



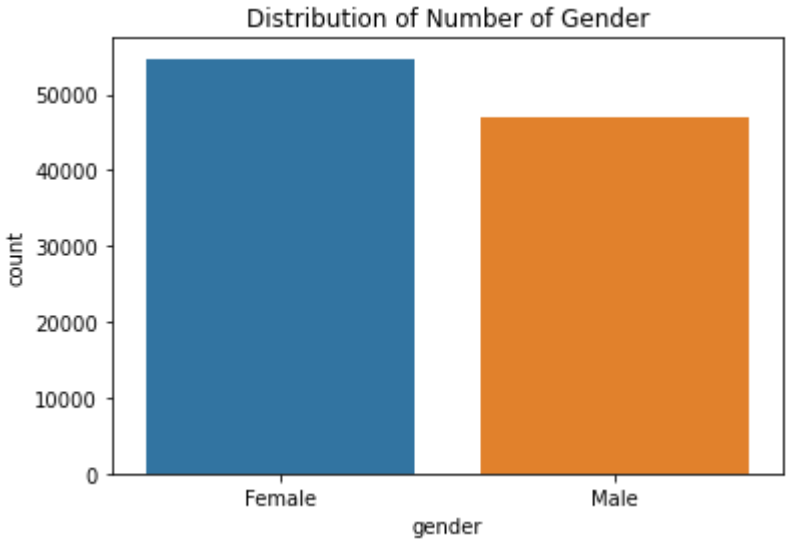
Most of the patients are Caucasian, followed by African Americans.

Although the Other values are few than Caucasian, we see that the Readmitted Probability almost close to Caucasian.

Gender

In [116…

```
sns.countplot(x = "gender", data = df)
plt.title("Distribution of Number of Gender")
plt.show()

print("Proportions of Race Value")
print(df.gender.value_counts(normalize = True))
```

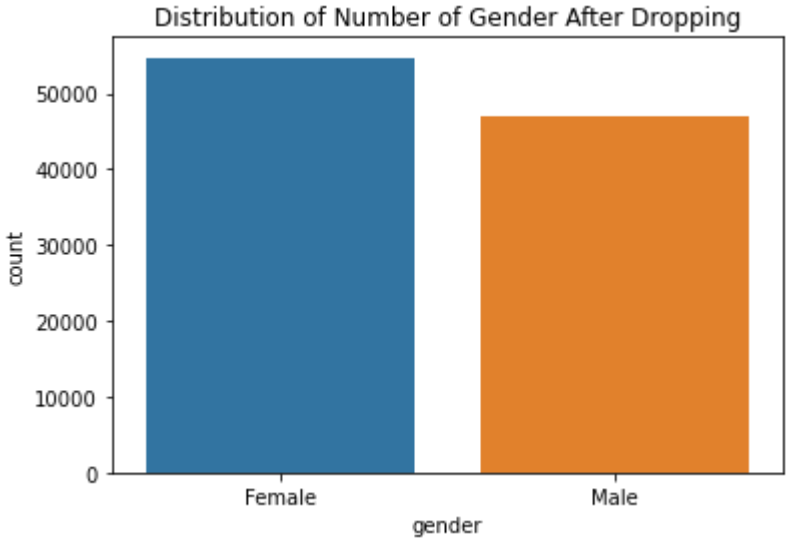


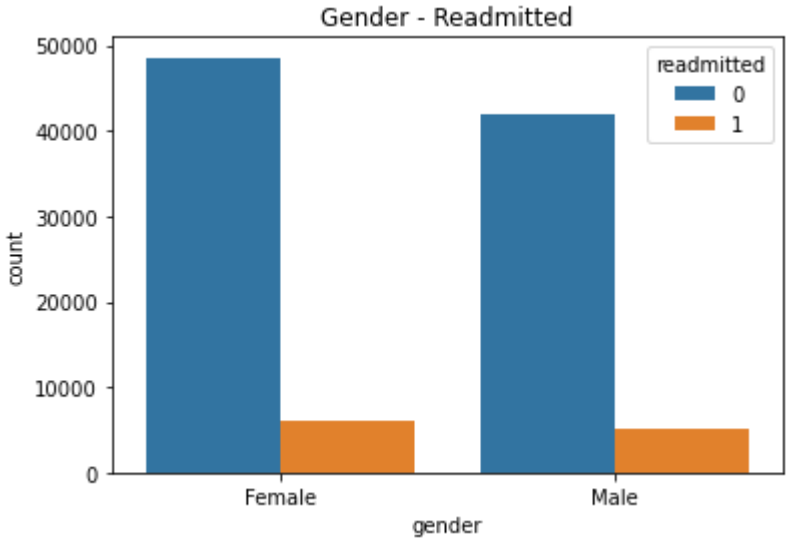

```
Proportions of Race Value
Female    0.537602
```

```
Male      0.462398
Name: gender, dtype: float64
```

In [117…

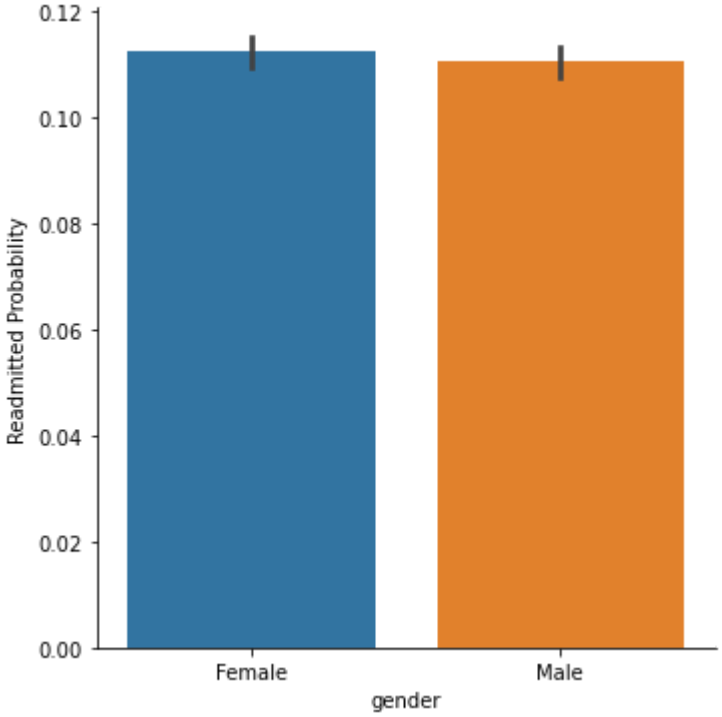
```
df = df.drop(df.loc[df["gender"]=="Unknown/Invalid"].index, axis=0)

sns.countplot(x = "gender", data = df)
plt.title("Distribution of Number of Gender After Dropping")
plt.show()

sns.countplot(x = "gender", hue = "readmitted", data = df)
plt.title("Gender - Readmitted")
plt.show()
```





In [118…

```
g = sns.catplot(x = "gender",y = "readmitted",
                data = df, kind = "bar", height= 5)
g.set_ylabels("Readmitted Probability")
plt.show()
```



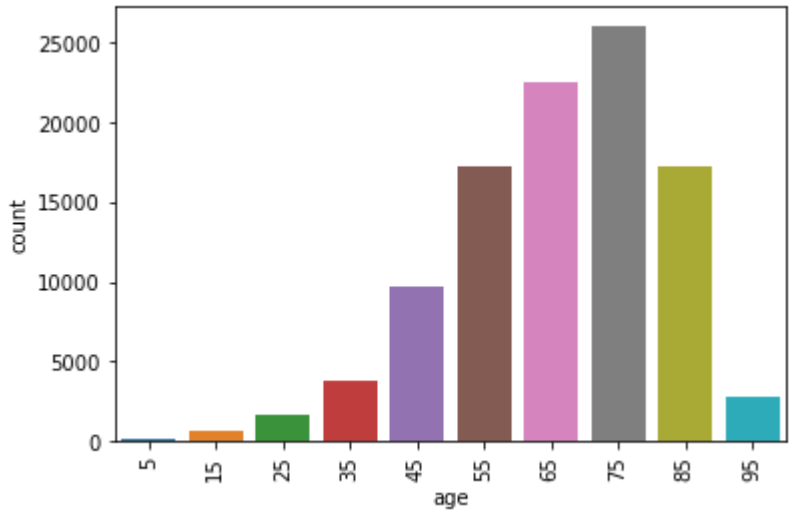
We see a nearly equal distribution of Gender.

Also, we can state that Females are a little more prone than Males.

Age

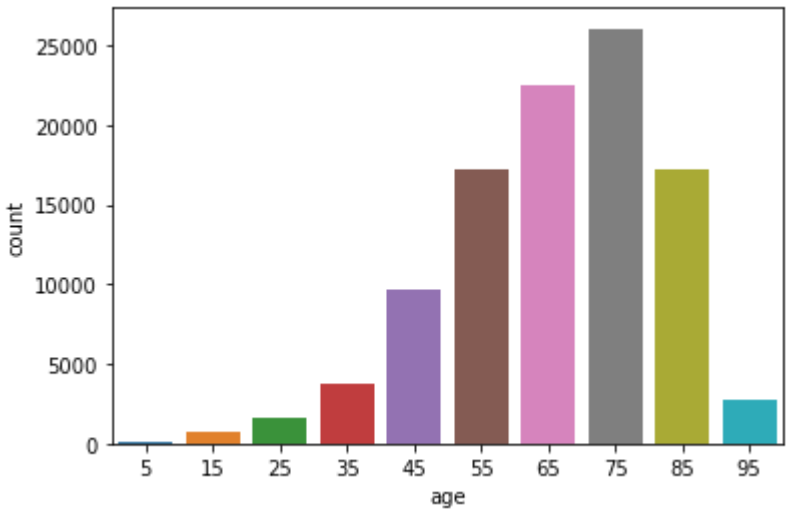
In [119…

```
sns.countplot(x="age", data = df)
plt.xticks(rotation = 90)
plt.show()
```



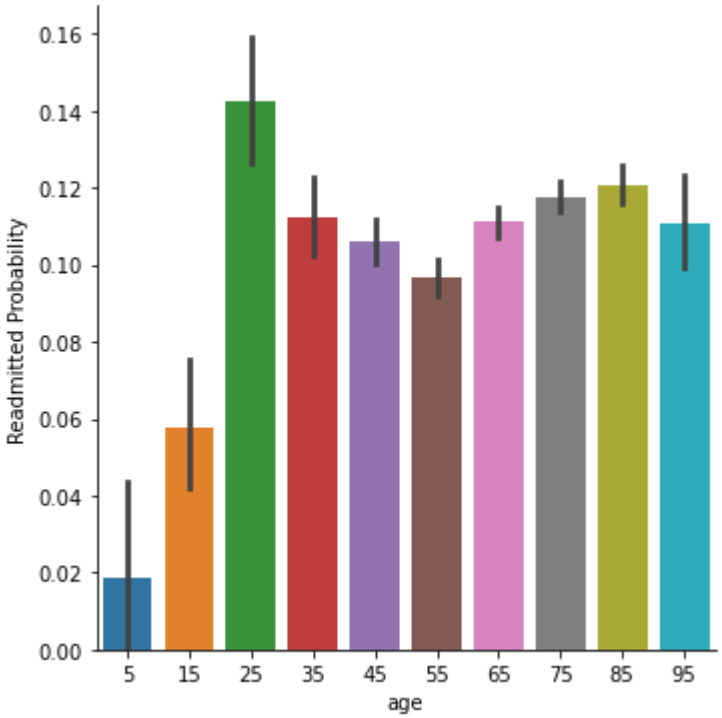
In [120…

```
sns.countplot(x="age", data = df)
#plt.xticks(rotation = 90)
plt.show()
```



In [121…

```
g = sns.catplot(x = "age", y = "readmitted", data = df,kind = "bar", height = 5)
g.set_ylabels("Readmitted Probability")
plt.show()
```



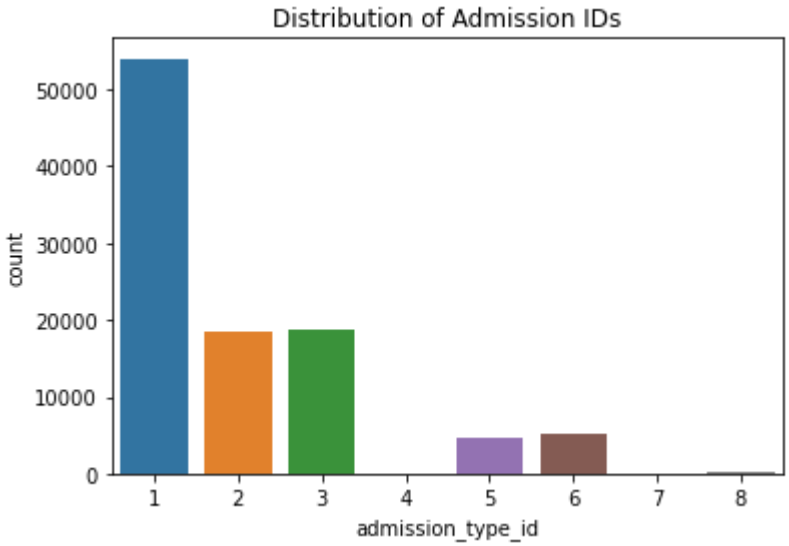
we can understand that we have an elderly population.

Weight

As with the age variable, we see the same situation for weight and decided to managed like below.

In [122…

```
sns.countplot(x = "admission_type_id", data = df)
plt.title("Distribution of Admission IDs")
plt.show()

print("Distribution of ID's")
print(df.admission_type_id.value_counts())
```




```
Distribution of ID's
1    53988
3    18868
2    18480
6     5291
5     4785
8      320
7       21
4       10
Name: admission_type_id, dtype: int64
```

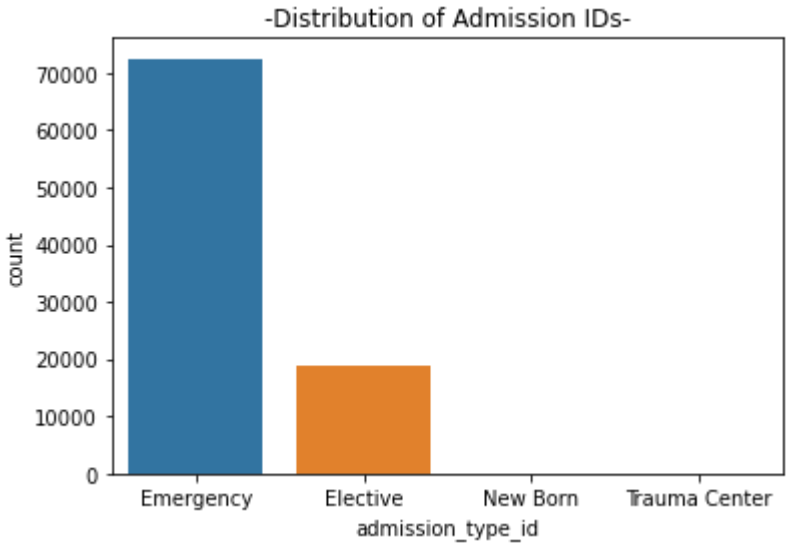
In here we need to do mapping for :

NULL, Not Available and Not Mapped values.

In addition, we will map Urgent value as Emergency because they have same meaning

In [123…

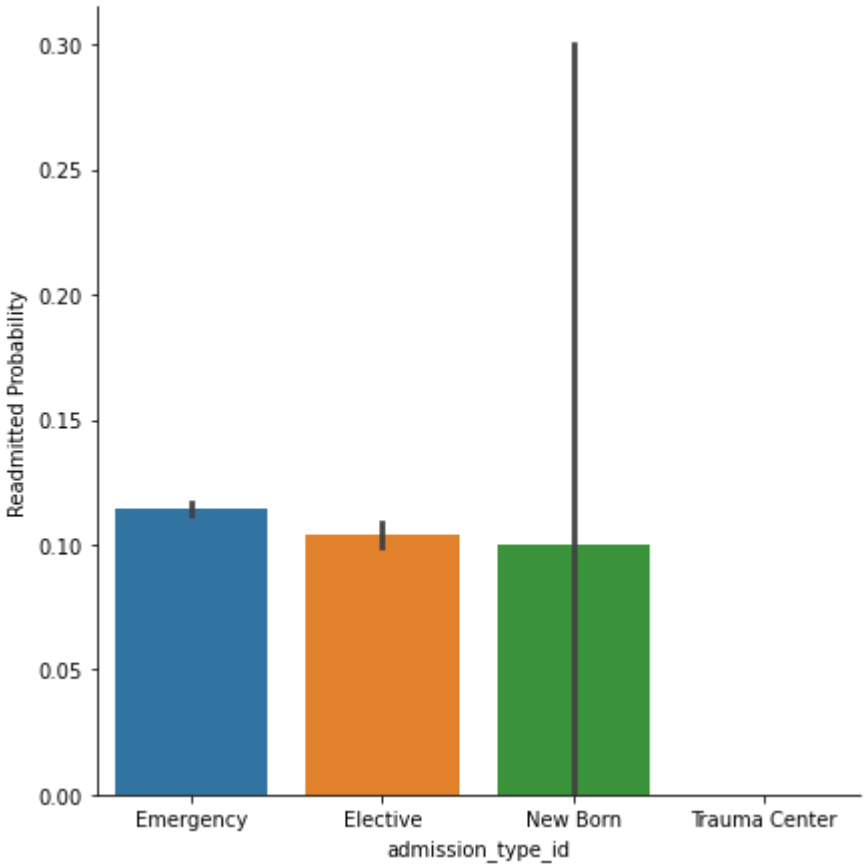
```
mapped = {1.0:"Emergency",
          2.0:"Emergency",
          3.0:"Elective",
          4.0:"New Born",
          5.0:np.nan,
          6.0:np.nan,
          7.0:"Trauma Center",
          8.0:np.nan}

df.admission_type_id = df.admission_type_id.replace(mapped)

sns.countplot(x = "admission_type_id", data = df)
plt.title("-Distribution of Admission IDs-")
plt.show()

print("-Distribution of ID's-")
print(df.admission_type_id.value_counts())
```




```
-Distribution of ID's-
Emergency        72468
Elective         18868
Trauma Center       21
New Born            10
Name: admission_type_id, dtype: int64
```

In [124…

```
g = sns.catplot(x = "admission_type_id", y ="readmitted",data = df, height = 6, kind =
g.set_ylabels("Readmitted Probability")
plt.show()
```

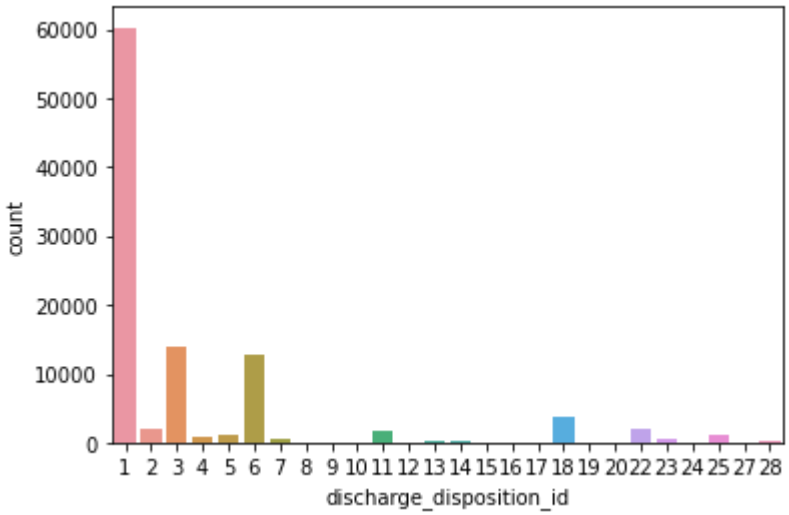


Discharge Disposition ID

-Integer identifier corresponding to 29 distinct values. For example, discharged to home, expired, and not available

In [125…

```
sns.countplot(x ="discharge_disposition_id", data = df)
plt.show()
```



When we look the graph, we can see too much values. Getting rid of that situation, we applied this rules:

If any one includes "home" word I will grouping into one

If not, it will be as OTHER

NAN = 18, 25, 26

In [126…

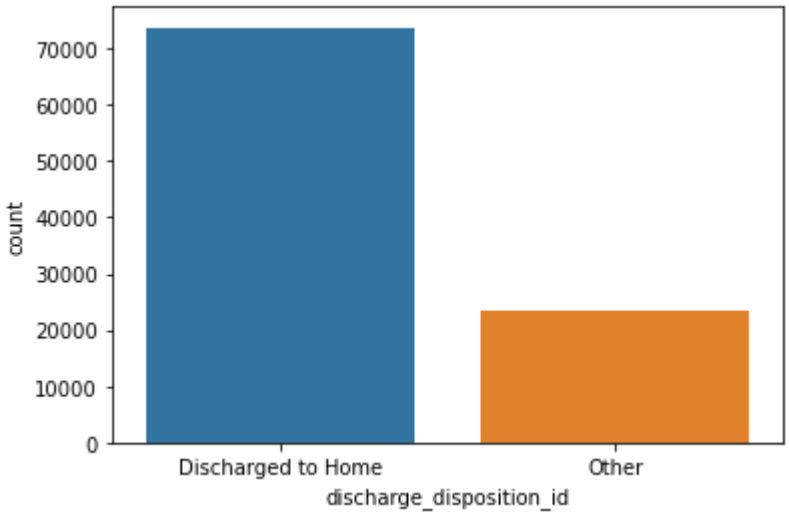
```
mapped_discharge = {1:"Discharged to Home",
                    6:"Discharged to Home",
                    8:"Discharged to Home",
                    13:"Discharged to Home",
                    19:"Discharged to Home",
                    18:np.nan,25:np.nan,26:np.nan,
                    2:"Other",3:"Other",4:"Other",
                    5:"Other",7:"Other",9:"Other",
                    10:"Other",11:"Other",12:"Other",
                    14:"Other",15:"Other",16:"Other",
                    17:"Other",20:"Other",21:"Other",
                    22:"Other",23:"Other",24:"Other",
                    27:"Other",28:"Other",29:"Other",30:"Other"}

df["discharge_disposition_id"] = df["discharge_disposition_id"].replace(mapped_discharg
```

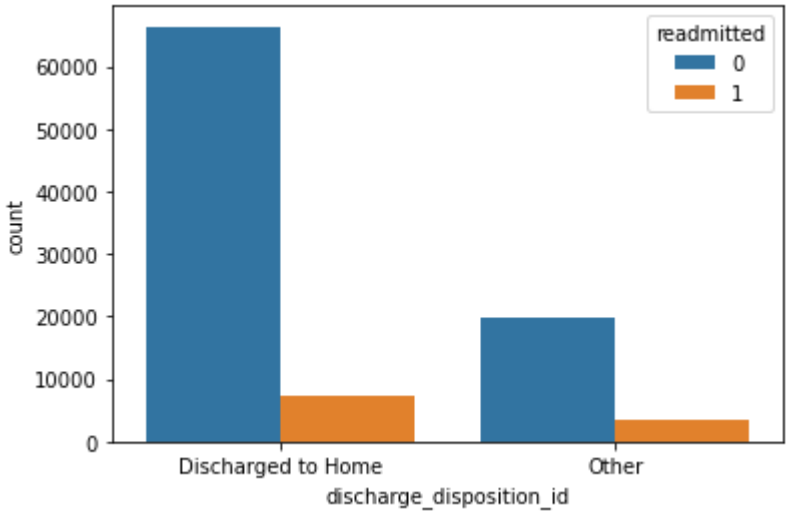
Now it will be more clear and readable

In [127…

```
sns.countplot(x ="discharge_disposition_id", data = df)
plt.show()

sns.countplot(x ="discharge_disposition_id", hue = "readmitted", data = df)
plt.show()
```

```
print("Proportions of ID's")
print(df.discharge_disposition_id.value_counts())
```

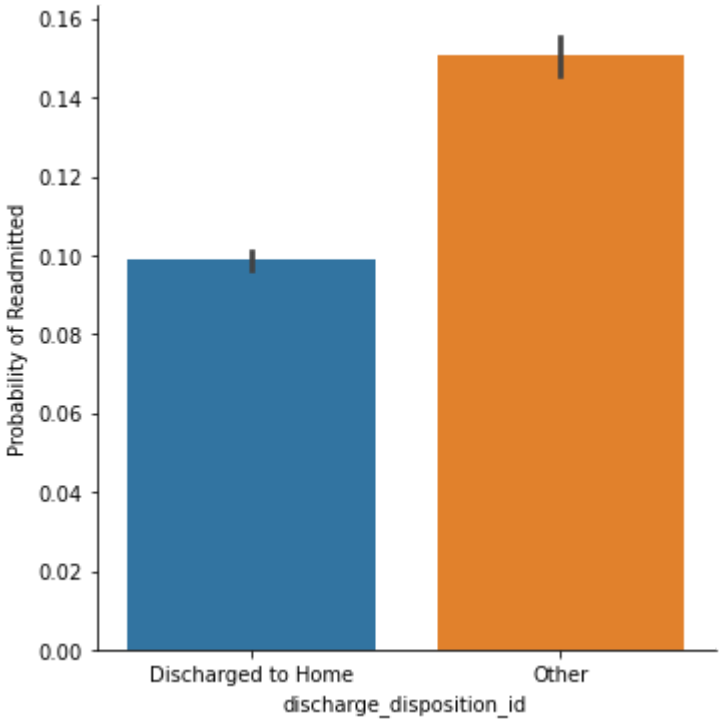





```
Proportions of ID's
Discharged to Home    73649
Other                 23434
Name: discharge_disposition_id, dtype: int64
```

In [128…

```
g = sns.catplot(x = "discharge_disposition_id", y="readmitted",data = df, height = 5, k
g.set_ylabels("Probability of Readmitted")
plt.show()
```

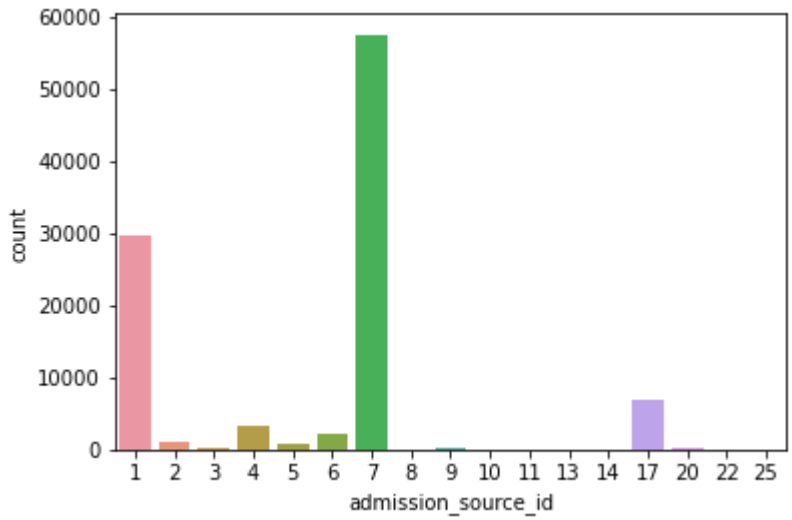


Admission Source ID

Integer identifier corresponding to 21 distinct values.For example, physician referral, emergency room, and transfer from a hospital

In [129…

```
sns.countplot(x ="admission_source_id", data = df)
plt.show()
```



We can see that there is same problem here. Again we applied some map like:

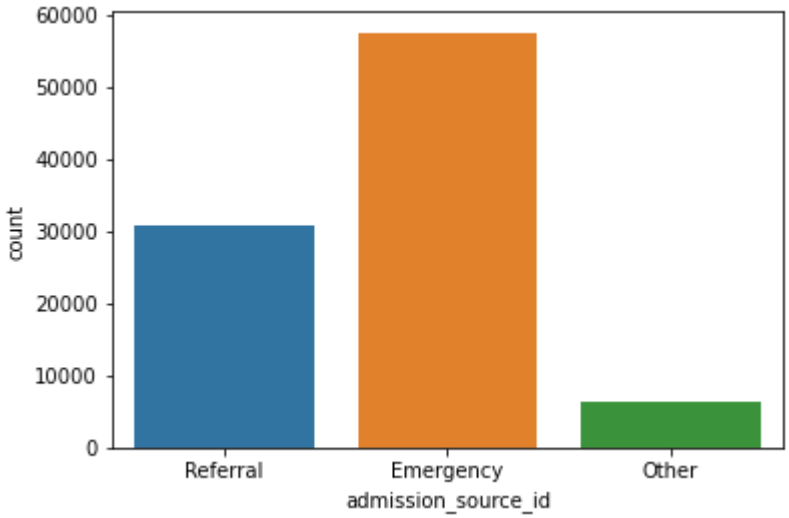
we'll put the similar ones together like Referral or Transfer

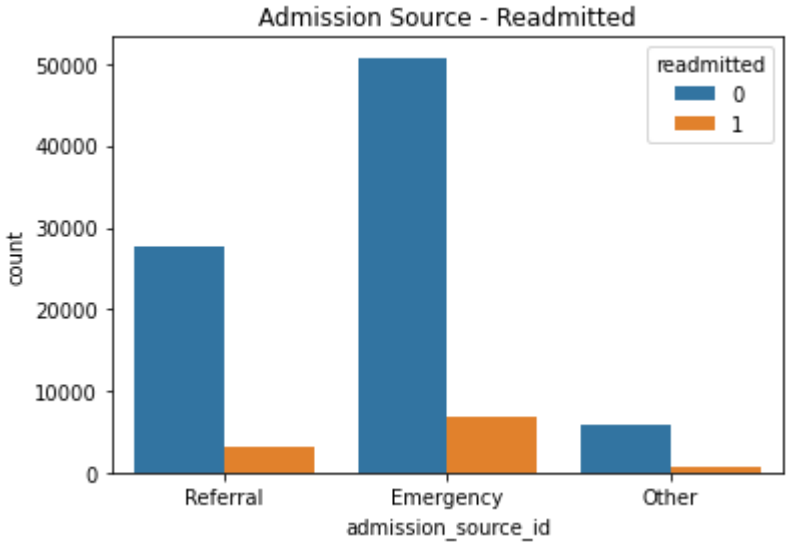
we will replace Null, Not Mapped, Unknown values as NAN

In [130…

```
mapped_adm = {1:"Referral",2:"Referral",3:"Referral",
              4:"Other",5:"Other",6:"Other",10:"Other",22:"Other",25:"Other",
              9:"Other",8:"Other",14:"Other",13:"Other",11:"Other",
```

```
                15:np.nan,17:np.nan,20:np.nan,21:np.nan,
                7:"Emergency"}
df.admission_source_id = df.admission_source_id.replace(mapped_adm)

sns.countplot(x = "admission_source_id", data = df)
plt.show()

sns.countplot(x = "admission_source_id", hue = "readmitted", data = df)
plt.title("Admission Source - Readmitted")
plt.show()


print(df.admission_source_id.value_counts())
```

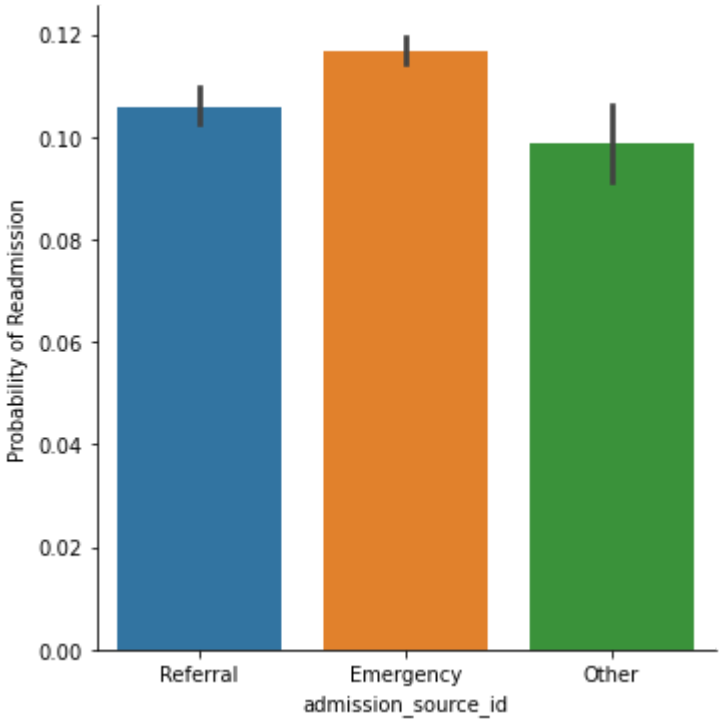





```
Emergency    57492
Referral     30855
Other         6474
Name: admission_source_id, dtype: int64
```

In [131…

```
g = sns.catplot(x = "admission_source_id", y ="readmitted",data = df, kind = "bar", hei
g.set_ylabels("Probability of Readmission")
plt.show()
```



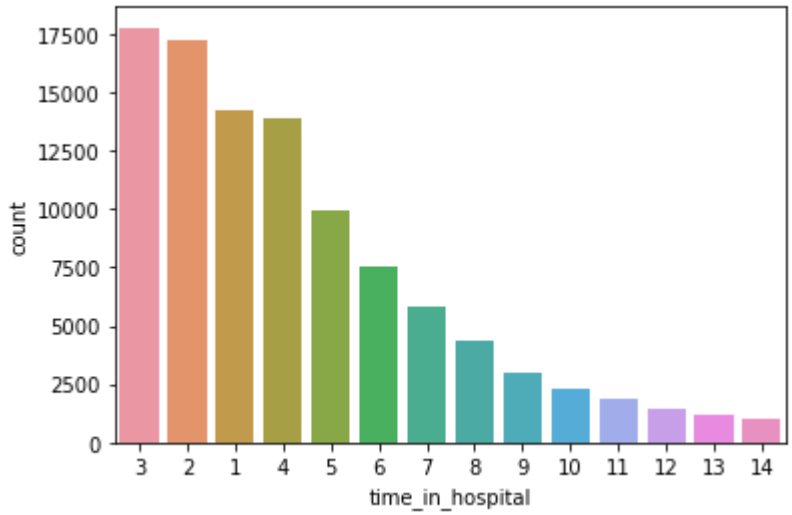
We see that Readmitted Probability of Referral is very close to Emergency, although Emergency is have more samples than other

Time in Hospital

Integer number of days between admission and discharge. Shortly it is "treatment time"

In [132…

```
sns.countplot(x="time_in_hospital", data = df,
              order = df.time_in_hospital.value_counts().index)
plt.show()

print(df.time_in_hospital.value_counts())
```

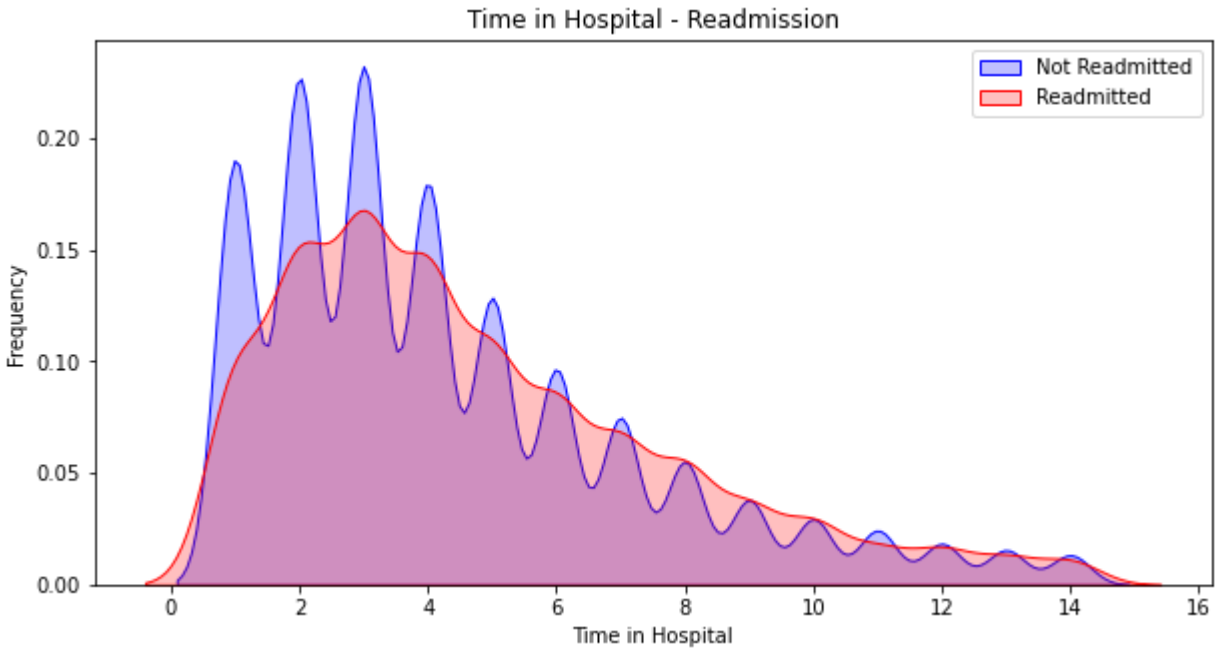



```
3     17756
2     17224
1     14206
4     13924
5      9966
6      7539
```

```
7      5859
8      4390
9      3002
10     2342
11     1855
12     1448
13     1210
14     1042
Name: time_in_hospital, dtype: int64
```

In [133…

```
fig = plt.figure(figsize=(10,5))

#readmitted = 0
ax = sns.kdeplot(df.loc[(df.readmitted == 0), "time_in_hospital"],
                 color = "b", shade = True, label = "Not Readmitted")

ax = sns.kdeplot(df.loc[(df.readmitted == 1), "time_in_hospital"],
                 color = "r", shade = True, label = "Readmitted")
ax.legend(loc="upper right")

ax.set_xlabel("Time in Hospital")
ax.set_ylabel("Frequency")
ax.set_title("Time in Hospital - Readmission")
plt.show()
```



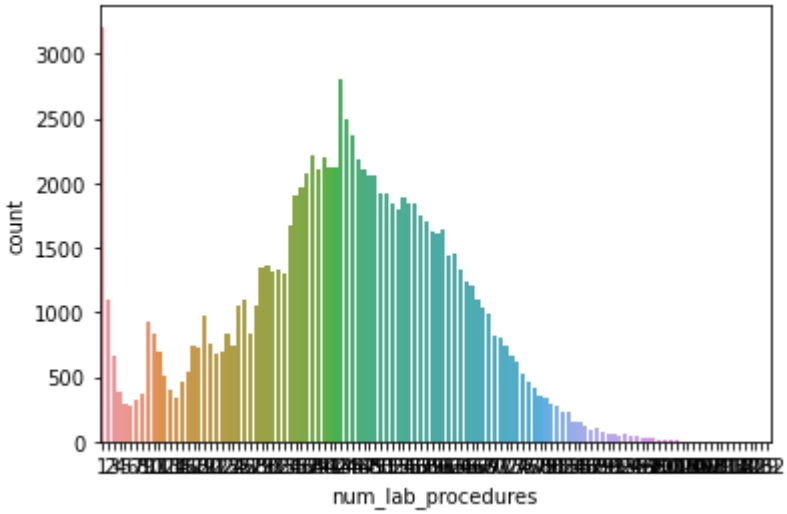
Medical Specialty Integer identifier of a specialty of the admitting physician, corresponding to 84 distinct values, for example, cardiology, internal medicine, family\general practice, and surgeon

We clearlt see that the Nephrology section has more probability than others

Number of Lab Procedures¶ Number of lab tests performed during the encounter

In [134…

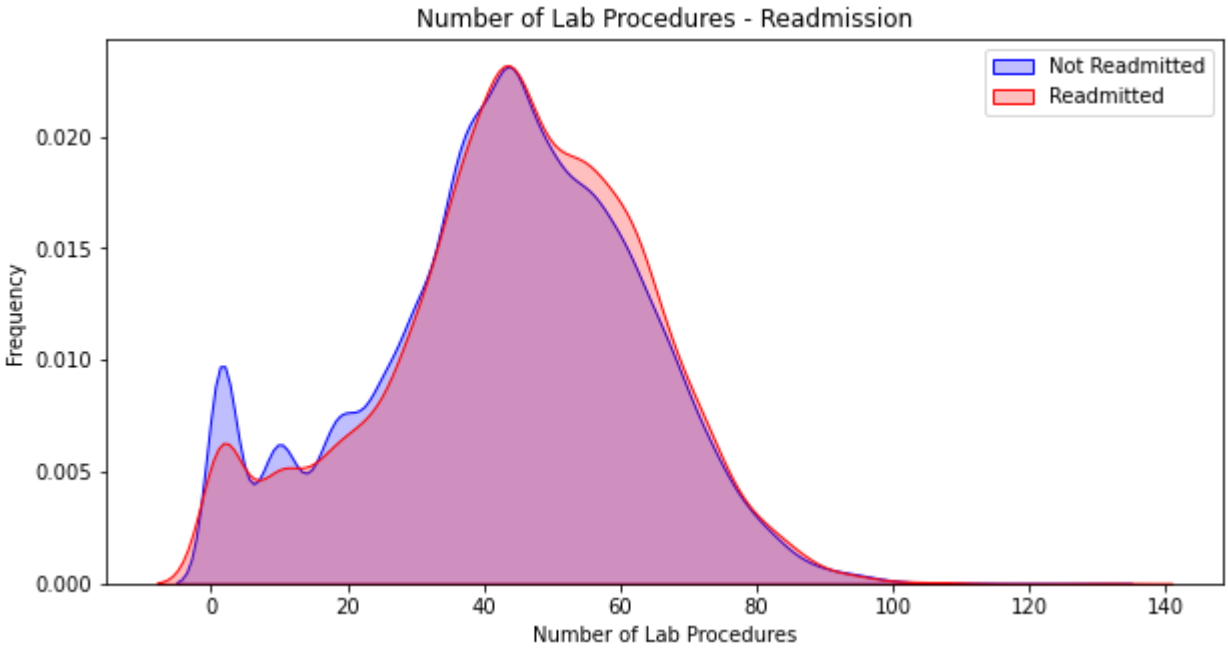
```
sns.countplot(x = "num_lab_procedures", data = df)
plt.show()

print("Proportions of Column")
print(df.num_lab_procedures.value_counts().head(10))
```




```
Proportions of Column
1     3208
43    2804
44    2496
45    2376
38    2212
40    2201
46    2189
41    2117
42    2113
47    2106
Name: num_lab_procedures, dtype: int64
```

In [135…

```
fig = plt.figure(figsize=(10,5))

#readmitted = 0
ax = sns.kdeplot(df.loc[(df.readmitted == 0), "num_lab_procedures"],
                 color = "b", shade = True,label = "Not Readmitted")

#readmitted = 1
ax = sns.kdeplot(df.loc[(df.readmitted == 1), "num_lab_procedures"],
                 color = "r", shade = True, label = "Readmitted")

ax.legend(loc="upper right")

ax.set_xlabel("Number of Lab Procedures")
ax.set_ylabel("Frequency")
ax.set_title("Number of Lab Procedures - Readmission")

plt.show()
```

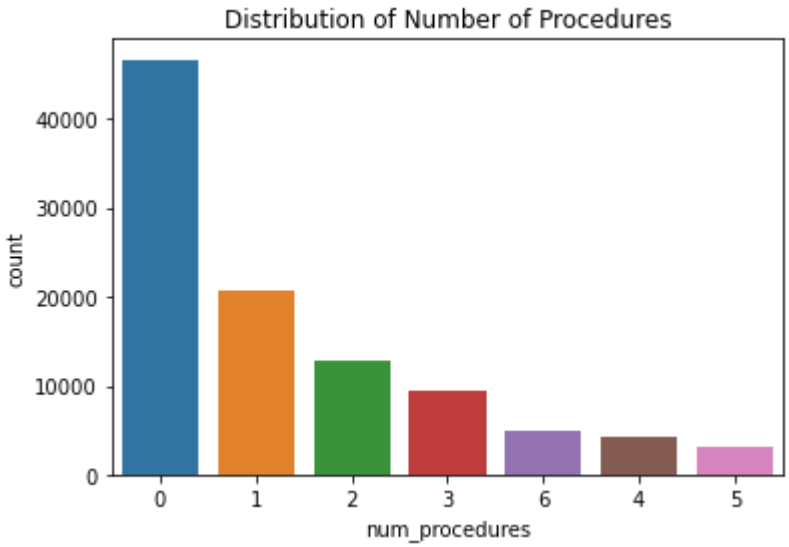


Number of Procedures

Number of procedures (other than lab tests) performed during the encounter

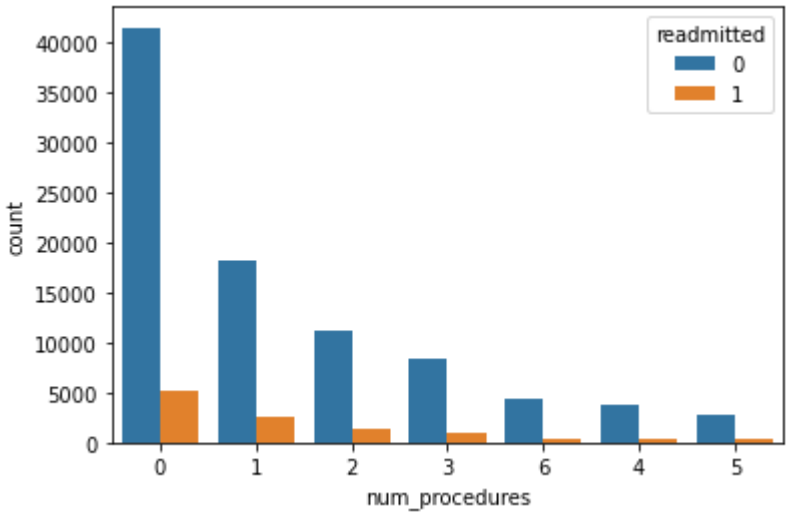
In [136…

```
sns.countplot(x = df.num_procedures, order = df.num_procedures.value_counts().index)
plt.title("Distribution of Number of Procedures")
plt.show()

print("Proportions of Values")
print(df.num_procedures.value_counts(normalize=True)*100)
```

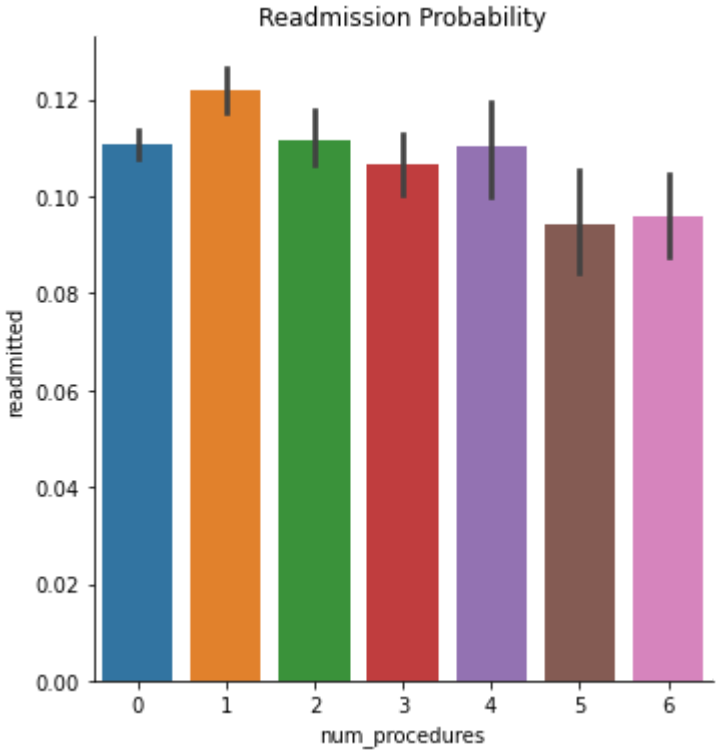



```
Proportions of Values
0    45.843774
1    20.381671
2    12.495701
3     9.279404
6     4.868174
4     4.107583
5     3.023692
Name: num_procedures, dtype: float64
```

In [137…
```
sns.countplot(x = "num_procedures", hue = "readmitted",
              data = df, order = df.num_procedures.value_counts().index)
plt.show()
```



In [138…
```
sns.catplot(x = "num_procedures", y = "readmitted",
             data = df, kind = "bar", height = 5)
plt.title("Readmission Probability")
plt.show()
```

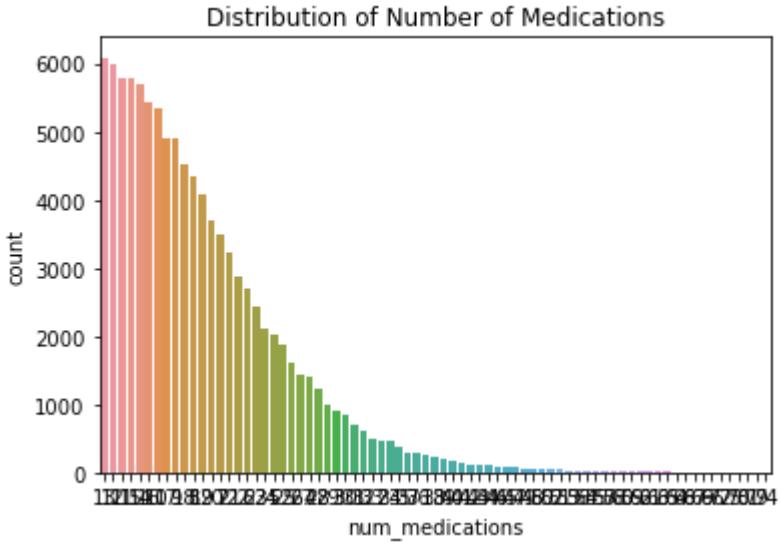


Number of Medications

Number of distinct generic names administered during the encounter

In [139…
```
sns.countplot(x="num_medications", data = df,
        order = df.num_medications.value_counts().index)
plt.title("Distribution of Number of Medications")
```

```
plt.show()

print(df.num_medications.value_counts())
```




```
13     6086
12     6004
11     5795
15     5792
14     5707
       ...
75        2
70        2
81        1
79        1
74        1
Name: num_medications, Length: 75, dtype: int64
```

Diag1, Diag2 and Diag3

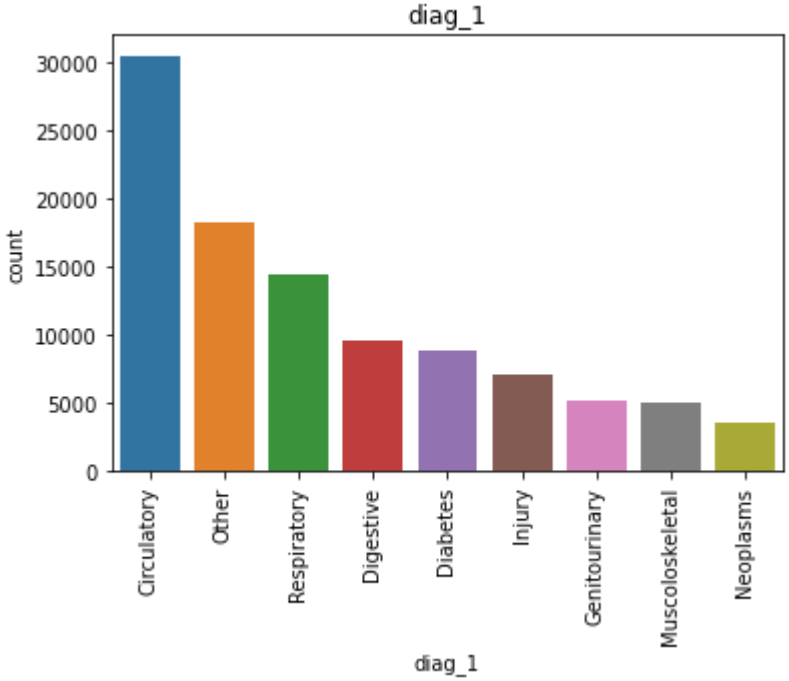
In diag section there are lots of ID that belong the specific name. So we'll map them

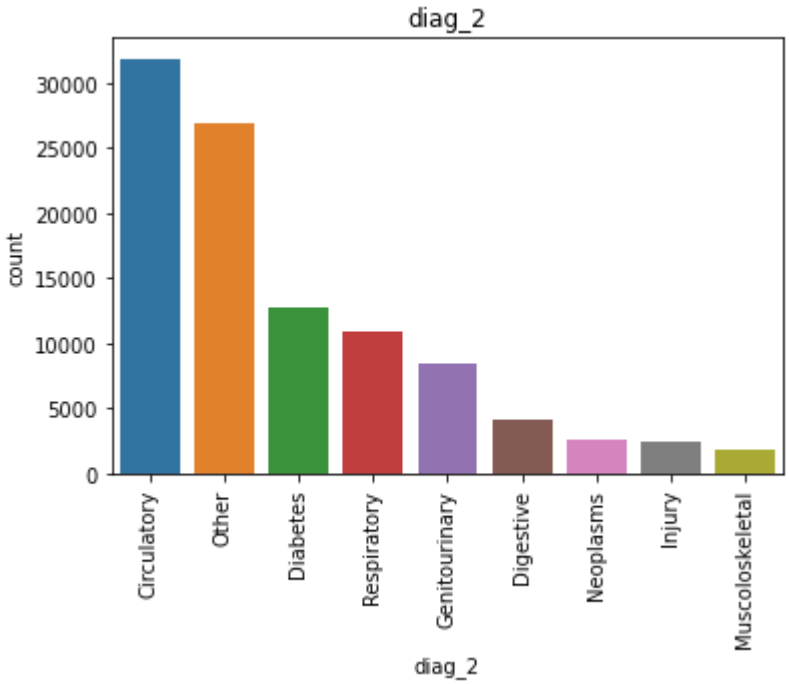
In [140…

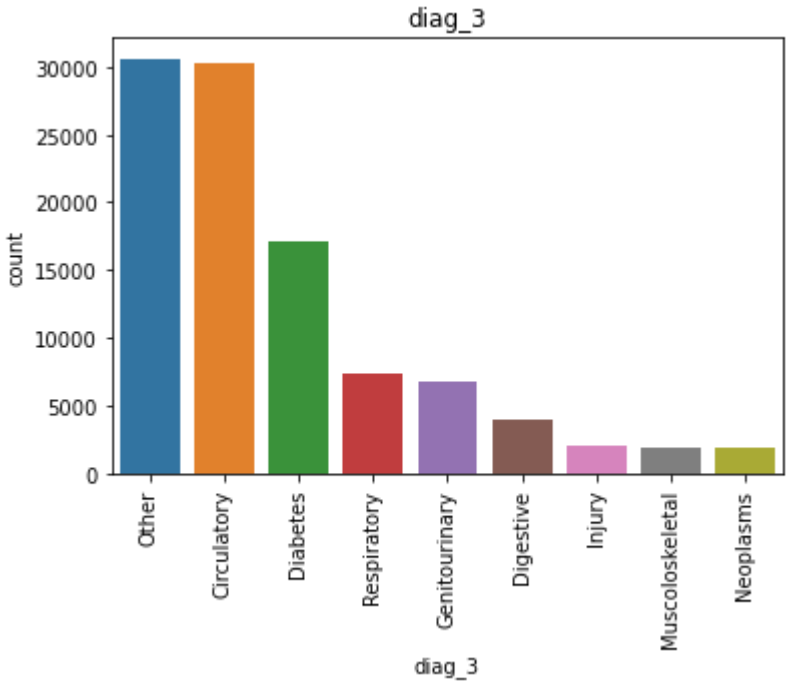
```
def map_diagnosis(data, cols):
    for col in cols:
        data.loc[(data[col].str.contains("V")) | (data[col].str.contains("E")), col] =
        data[col] = data[col].astype(np.float16)

    for col in cols:
        data["temp_diag"] = np.nan
        data.loc[(data[col]>=390) & (data[col]<=459) | (data[col]==785), "temp_diag"] =
        data.loc[(data[col]>=460) & (data[col]<=519) | (data[col]==786), "temp_diag"] =
        data.loc[(data[col]>=520) & (data[col]<=579) | (data[col]==787), "temp_diag"] =
        data.loc[(data[col]>=250) & (data[col]<251), "temp_diag"] = "Diabetes"
        data.loc[(data[col]>=800) & (data[col]<=999), "temp_diag"] = "Injury"
        data.loc[(data[col]>=710) & (data[col]<=739), "temp_diag"] = "Muscoloskeletal"
        data.loc[(data[col]>=580) & (data[col]<=629) | (data[col] == 788), "temp_diag"]
        data.loc[(data[col]>=140) & (data[col]<=239), "temp_diag"] = "Neoplasms"

        data["temp_diag"] = data["temp_diag"].fillna("Other")
        data[col] = data["temp_diag"]
        data = data.drop("temp_diag", axis=1)

    return data
```

In [141…
```
df = map_diagnosis(df,["diag_1","diag_2","diag_3"])
```

In [142…
```
def plot_diags(col,data):
    sns.countplot(x = col, data = data,
            order = data[f"{col}"].value_counts().index)
    plt.xticks(rotation = 90)
    plt.title(col)
    plt.show()

diag_cols = ["diag_1","diag_2","diag_3"]

for diag in diag_cols:
    plot_diags(diag,df)
```





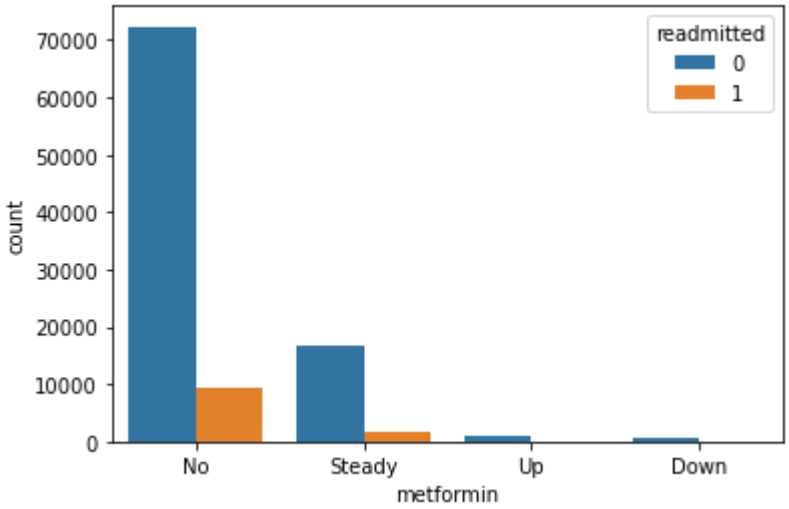


Diabetes medications

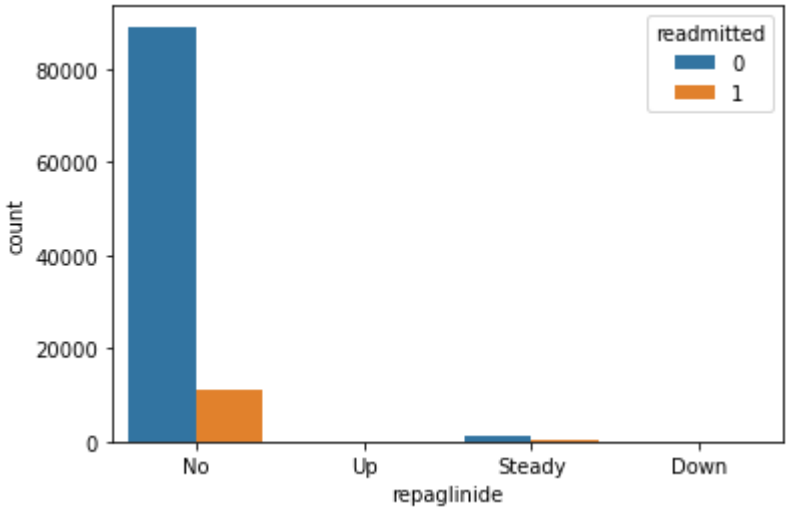
There was too many medications that belong the diabet. And some of them has just one or two value that does not any impact the model. So we decided to drop them. But firstly, lets look at the medications

In [143…

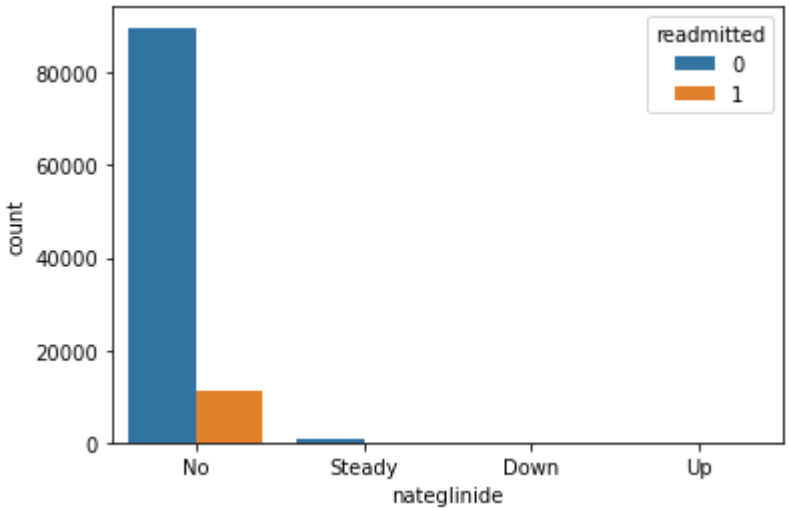
```
drug_cols = ['metformin', 'repaglinide', 'nateglinide', 'chlorpropamide',
             'glimepiride', 'acetohexamide', 'glipizide', 'glyburide', 'tolbutamide',
             'pioglitazone', 'rosiglitazone', 'acarbose', 'miglitol', 'troglitazone',
             'tolazamide', 'insulin', 'glyburide-metformin', 'glipizide-metformin',
             'metformin-rosiglitazone', 'metformin-pioglitazone']

def explore_drug(drugs):

    for drug in drugs:
        sns.countplot(x = drug,hue = "readmitted",data = df)
        plt.show()
        print(drug.upper())
        print(df[f"{drug}"].value_counts())


explore_drug(drug_cols)
```



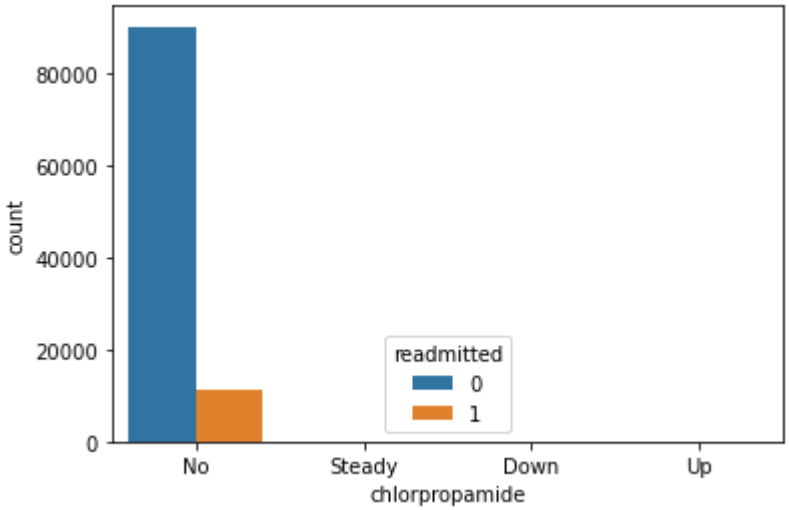

```
METFORMIN
No        81776
Steady    18345
Up         1067
Down        575
Name: metformin, dtype: int64
```



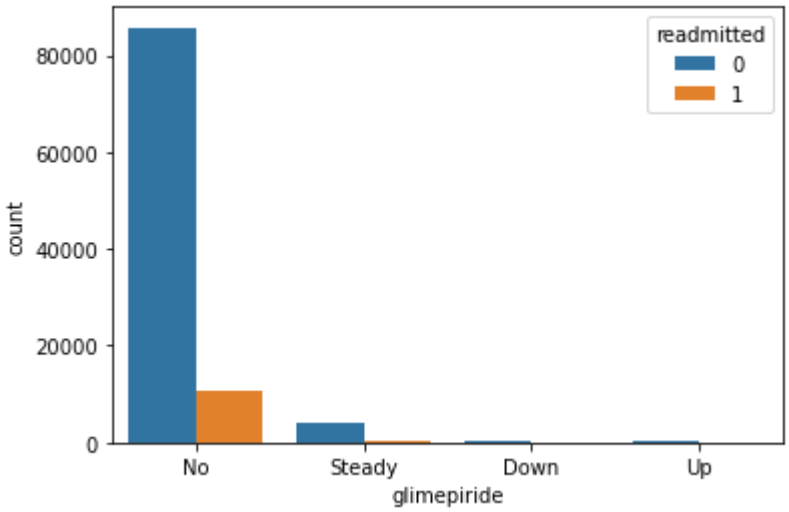

```
REPAGLINIDE
No        100224
Steady      1384
Up           110
Down          45
Name: repaglinide, dtype: int64
```



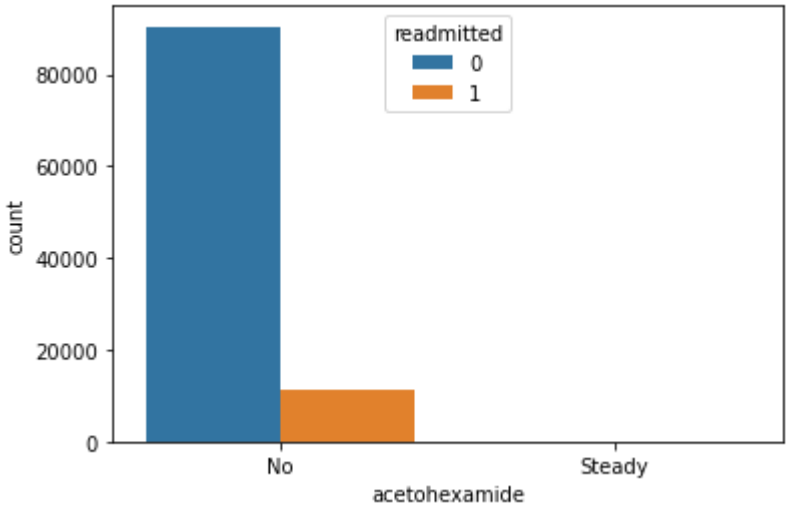
```
NATEGLINIDE
No        101060
Steady       668
Up            24
Down          11
Name: nateglinide, dtype: int64
```



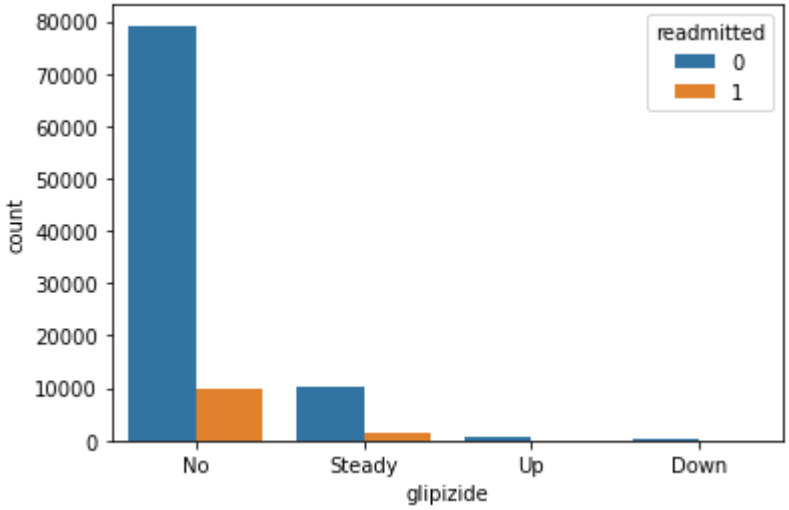

```
CHLORPROPAMIDE
No        101677
Steady        79
Up             6
Down           1
Name: chlorpropamide, dtype: int64
```



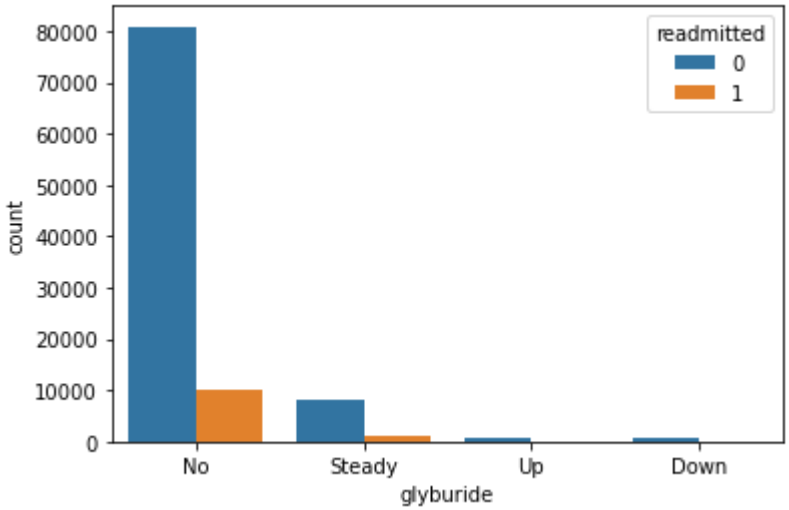

```
GLIMEPIRIDE
No        96572
Steady     4670
Up          327
Down        194
Name: glimepiride, dtype: int64
```




```
ACETOHEXAMIDE
No        101762
Steady         1
Name: acetohexamide, dtype: int64
```

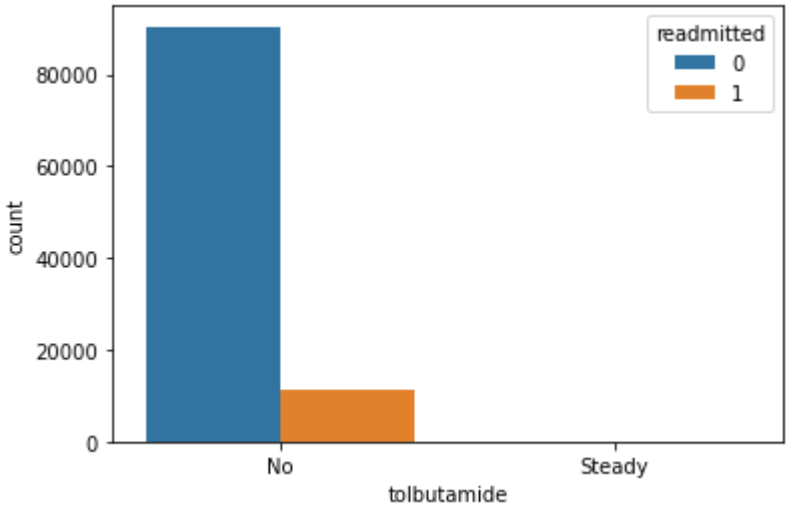



```
GLIPIZIDE
No        89078
Steady    11355
Up          770
Down        560
Name: glipizide, dtype: int64
```

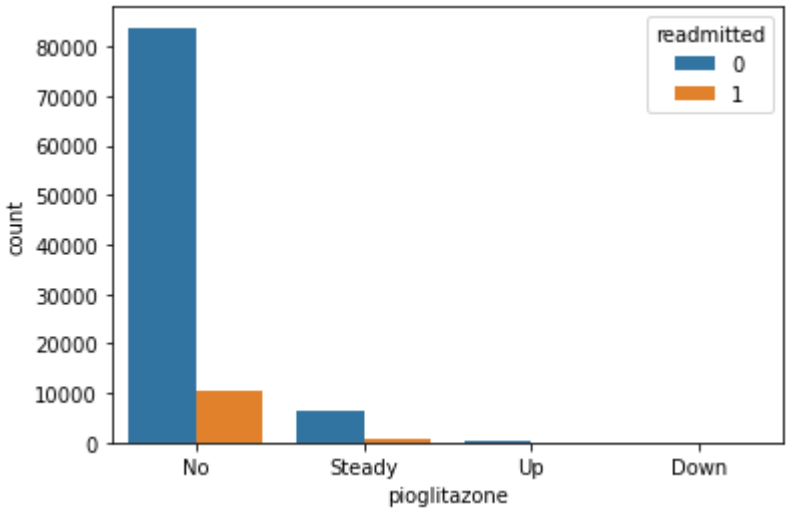



```
GLYBURIDE
No        91113
```

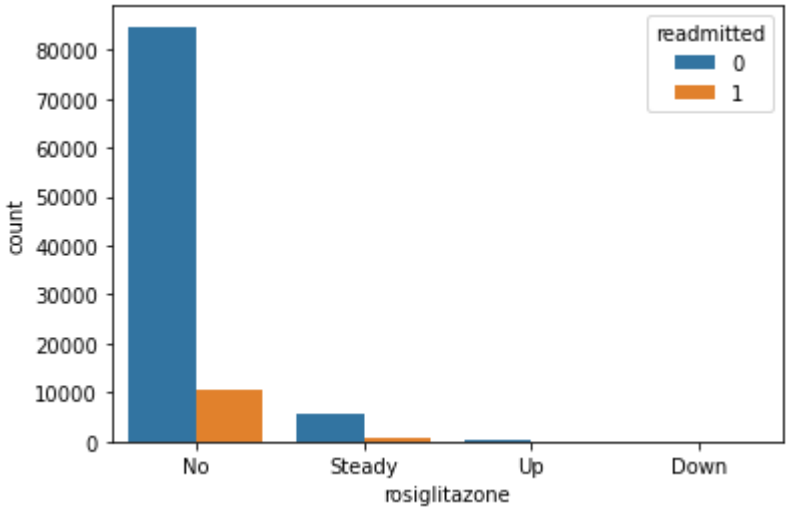
```
Steady     9274
Up          812
Down        564
Name: glyburide, dtype: int64
```



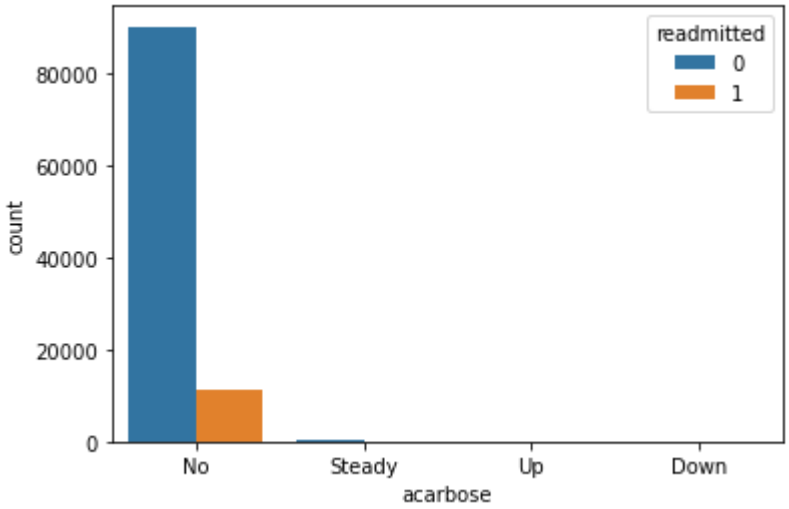

```
TOLBUTAMIDE
No        101740
Steady        23
Name: tolbutamide, dtype: int64
```



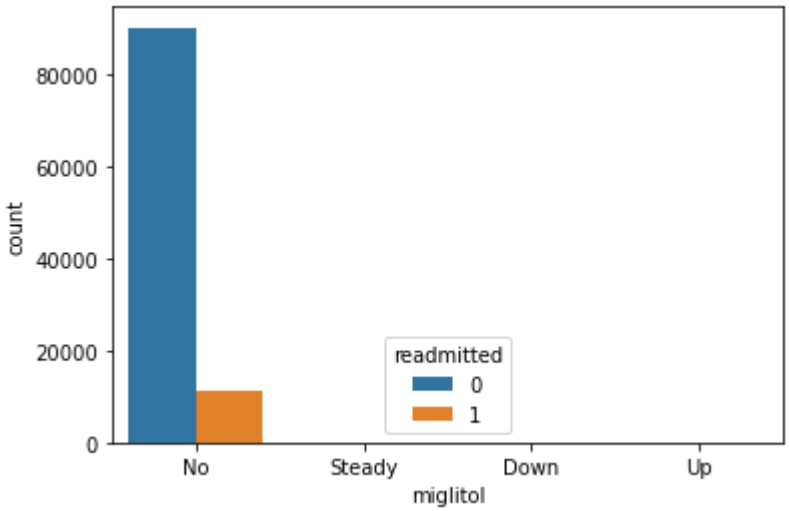

```
PIOGLITAZONE
No        94436
Steady     6975
Up          234
Down        118
Name: pioglitazone, dtype: int64
```



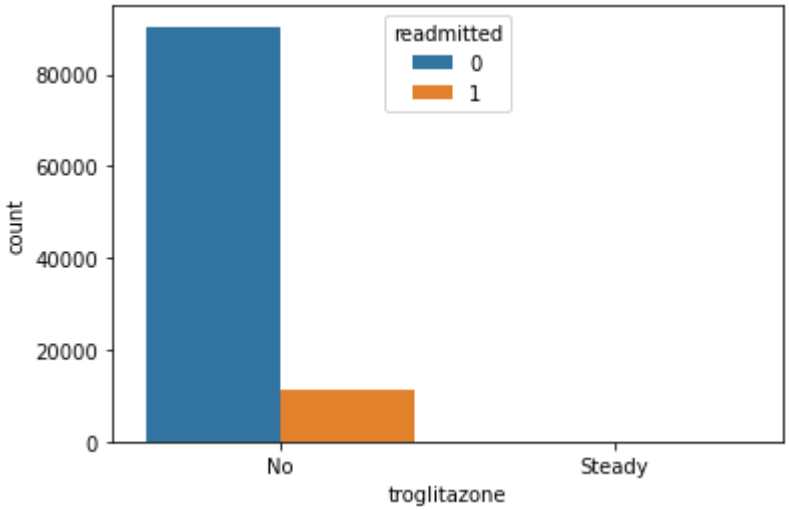

```
ROSIGLITAZONE
No        95399
Steady     6099
Up          178
Down         87
Name: rosiglitazone, dtype: int64
```



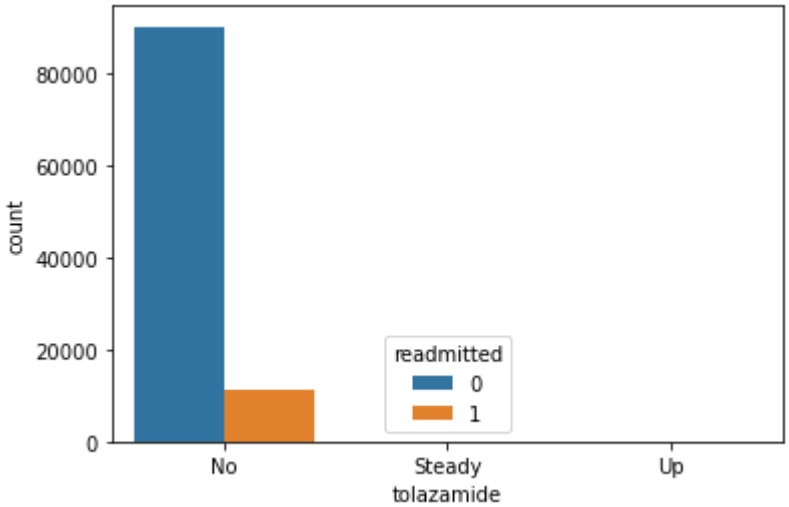

```
ACARBOSE
No        101455
Steady       295
Up            10
Down           3
Name: acarbose, dtype: int64
```



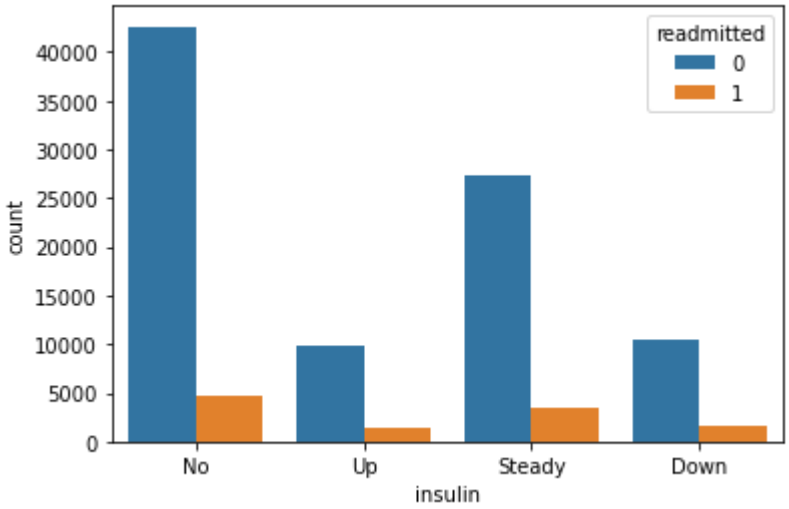
```
MIGLITOL
No        101725
Steady        31
Down           5
Up             2
Name: miglitol, dtype: int64
```



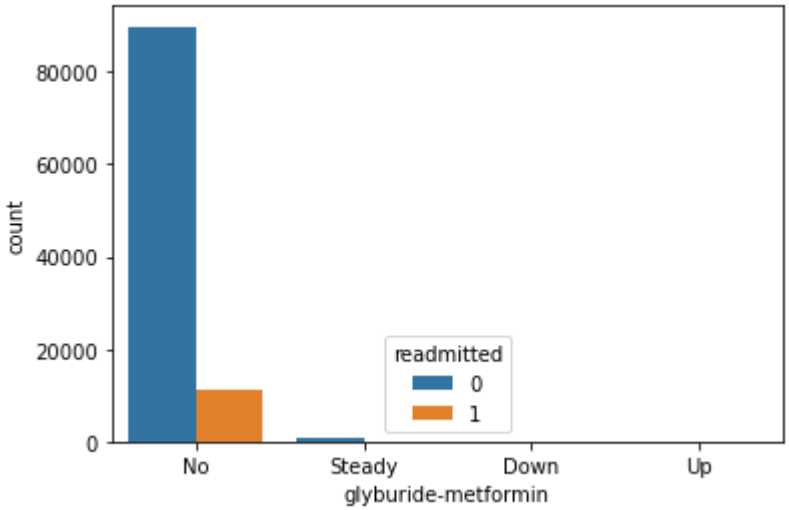

```
TROGLITAZONE
No        101760
Steady         3
Name: troglitazone, dtype: int64
```



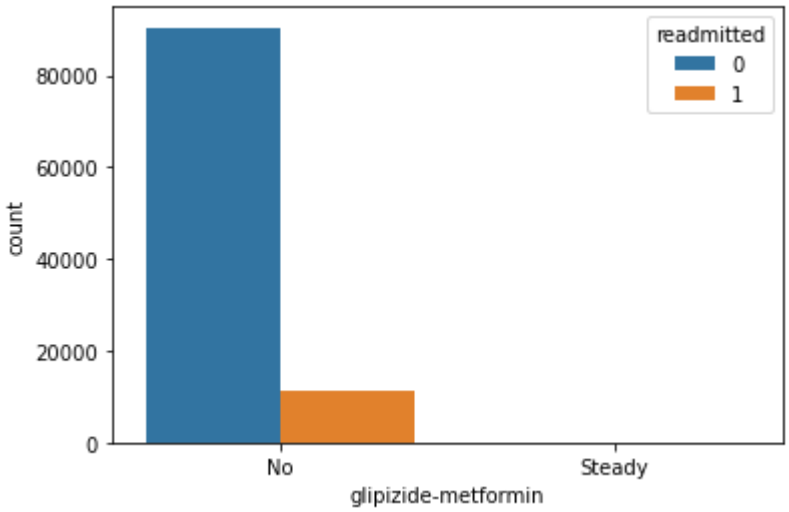

```
TOLAZAMIDE
No        101724
Steady        38
Up             1
Name: tolazamide, dtype: int64
```



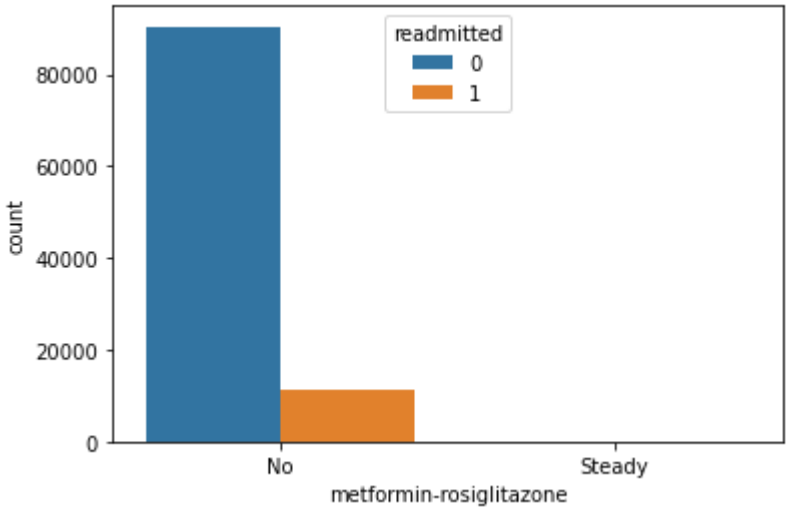

```
INSULIN
No        47380
Steady    30849
Down      12218
Up        11316
Name: insulin, dtype: int64
```



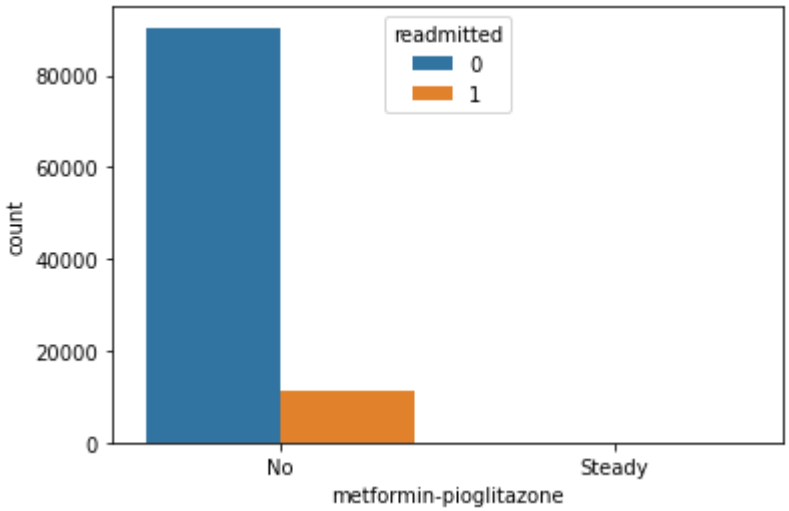

```
GLYBURIDE-METFORMIN
No        101057
Steady       692
Up             8
Down           6
Name: glyburide-metformin, dtype: int64
```



```
GLIPIZIDE-METFORMIN
No        101750
Steady        13
Name: glipizide-metformin, dtype: int64
```




```
METFORMIN-ROSIGLITAZONE
No        101761
Steady         2
Name: metformin-rosiglitazone, dtype: int64
```




```
METFORMIN-PIOGLITAZONE
No        101762
Steady         1
Name: metformin-pioglitazone, dtype: int64
```

Change

Indicates if there was a change in diabetic medications (either dosage or generic name). Values:

“change”

“no change”

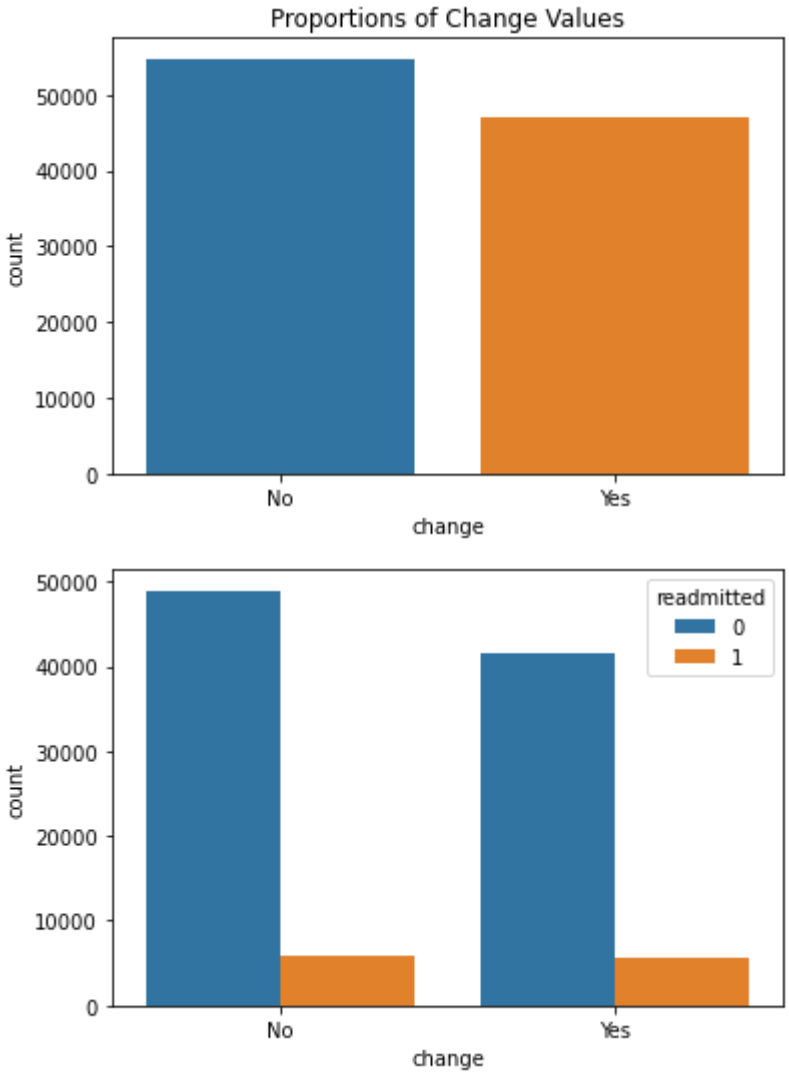
In [144…

```
df.change.value_counts()
```

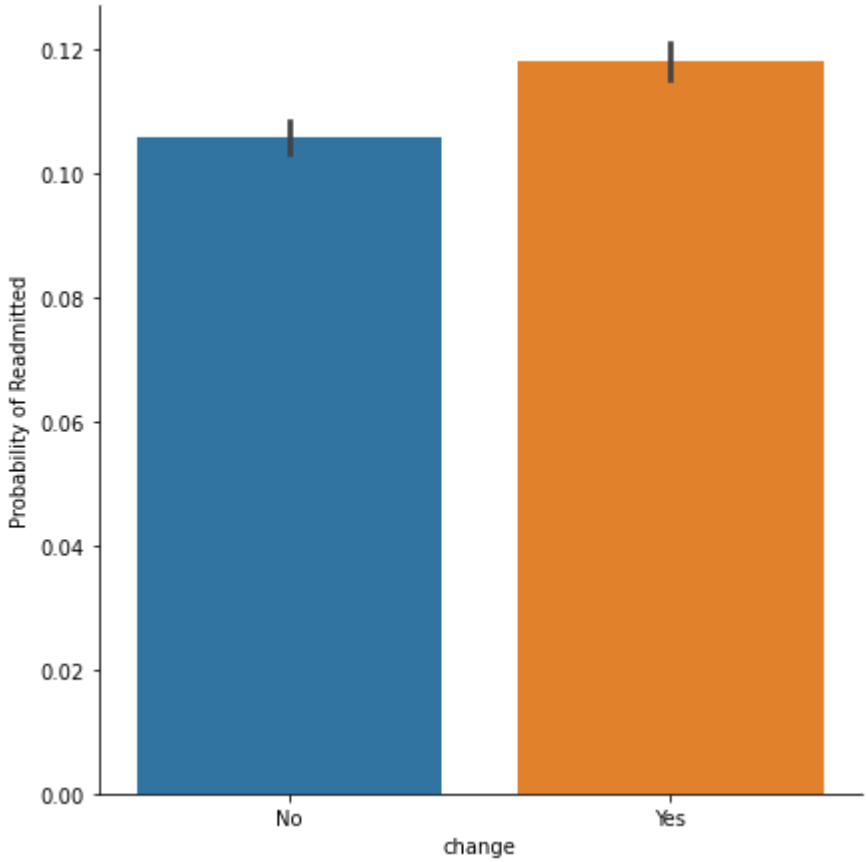
Out[144…

```
No    54754
Ch    47009
Name: change, dtype: int64
```

In [145…
```
df.change = df.change.replace("Ch","Yes")

sns.countplot(x = "change", data = df)
plt.title("Proportions of Change Values")
plt.show()

sns.countplot(x = "change", hue = "readmitted", data = df)
plt.show()
```



In [146…
```
g = sns.catplot(x = "change", y="readmitted", data = df, height = 6, kind ="bar")
g.set_ylabels("Probability of Readmitted")
plt.show()
```



Glucose Serum Test Result

Indicates the range of the result or if the test was not taken.

Values:

“>200,”

“>300,”

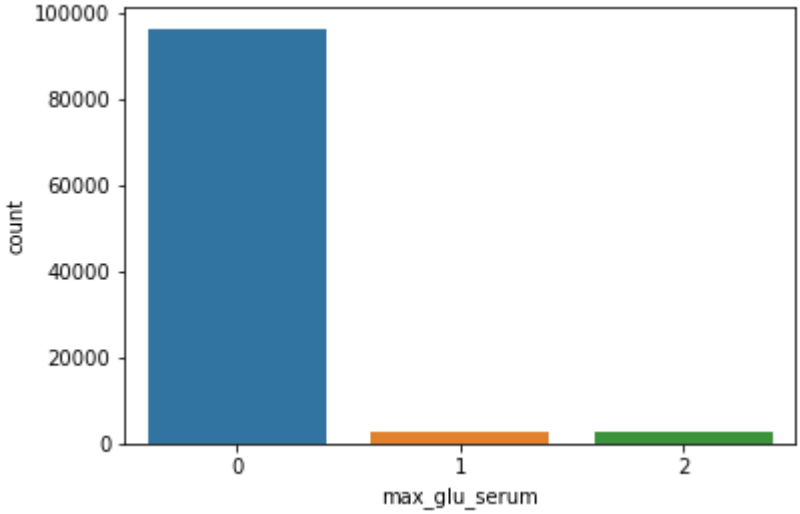
“normal,”

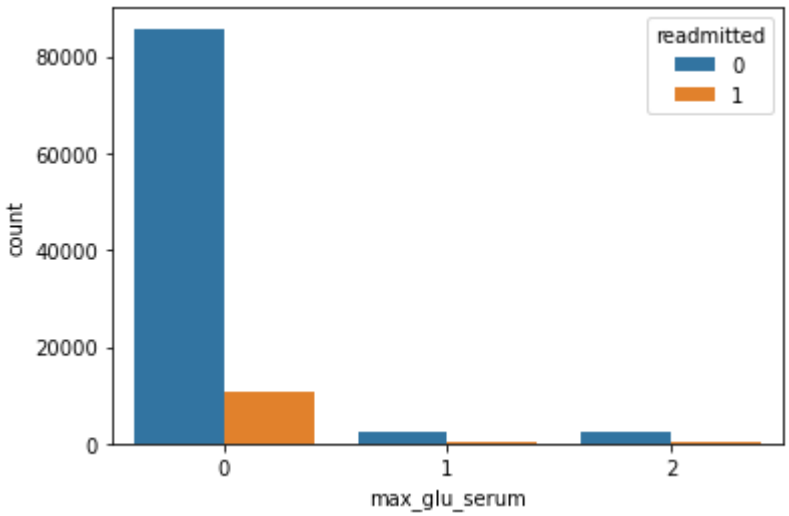
“none” if not measured

We decided use the Glucose Serum Test Result like as follows:

In [147…

```
df["max_glu_serum"] = df["max_glu_serum"].replace({">200":2,">300":2,"Norm":1,"None":0}

sns.countplot(x = "max_glu_serum", data = df)
plt.show()

sns.countplot(x = "max_glu_serum",hue = "readmitted", data = df)
plt.show()

print(df.max_glu_serum.value_counts())
```






```
0    96417
2     2749
1     2597
Name: max_glu_serum, dtype: int64
```

A1c test result

Indicates the range of the result or if the test was not taken.

Values:

">8" if the result was greater than 8%,

">7" if the result was greater than 7% but less than 8%, "normal"

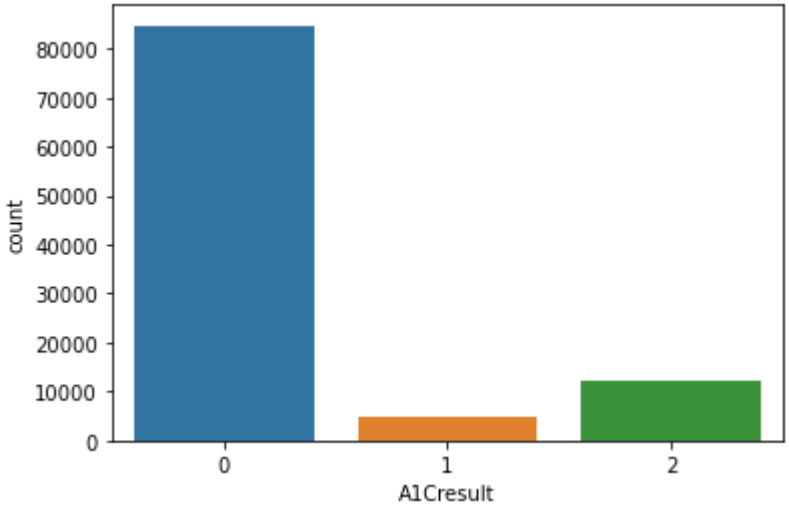
if the result was less than 7%, and "none" if not measured.

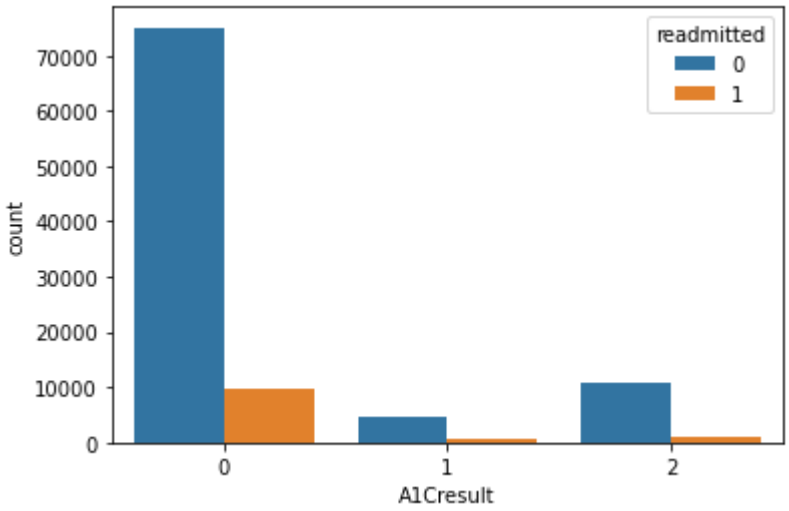
We decided use the A1c test result like as follows:

In [148…

```
df["A1Cresult"] = df["A1Cresult"].replace({">7":2,
                                           ">8":2,
                                           "Norm":1,
                                           "None":0})

sns.countplot(x = "A1Cresult", data = df)
plt.show()
```

```
sns.countplot(x = "A1Cresult",hue = "readmitted", data = df)
plt.show()

print(df.A1Cresult.value_counts())
```





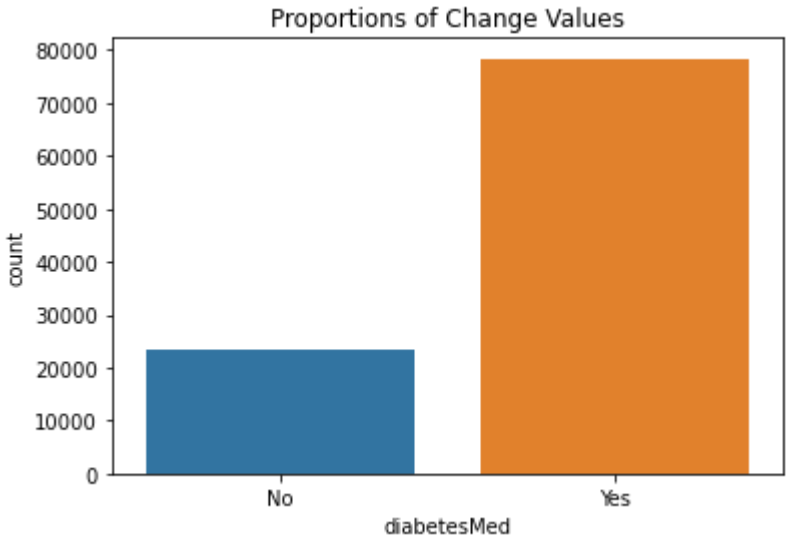

```
0    84745
2    12028
1     4990
Name: A1Cresult, dtype: int64
```

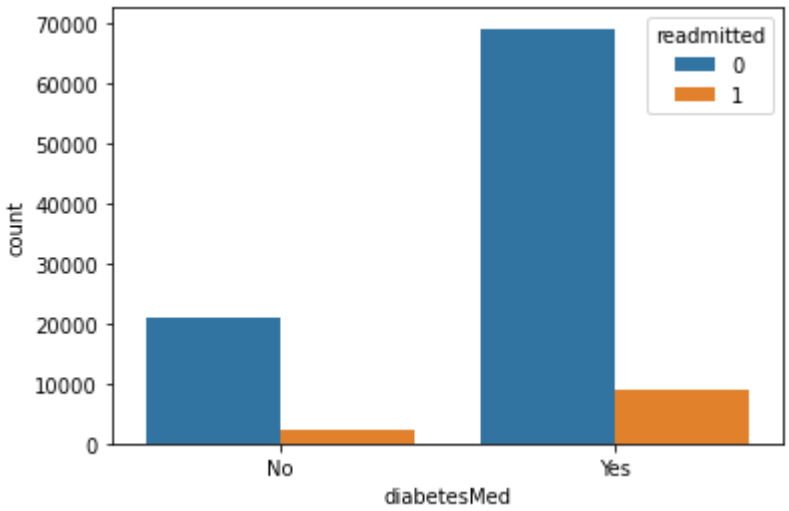
Diabetes medications

Indicates if there was any diabetic medication prescribed. Values: “yes” and “no”

In [149…

```
sns.countplot(x = "diabetesMed", data = df )
plt.title("Proportions of Change Values")
plt.show()

sns.countplot(x = "diabetesMed", hue = "readmitted", data = df)
plt.show()

print(df.diabetesMed.value_counts())
```

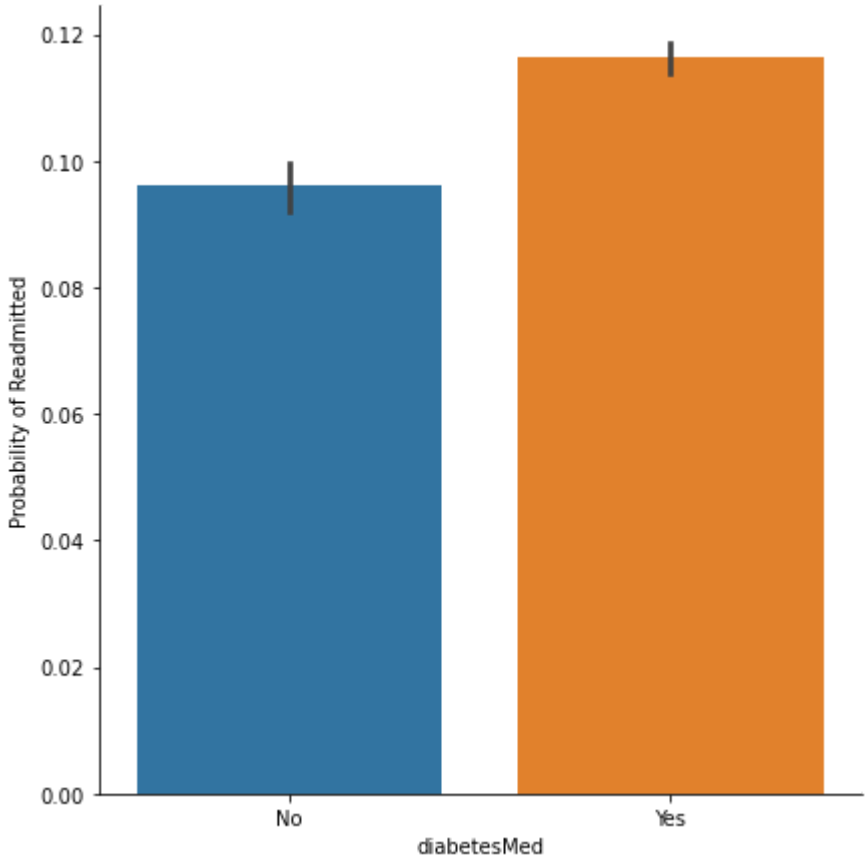





```
Yes    78361
No     23402
Name: diabetesMed, dtype: int64
```

In [150…

```
g = sns.catplot(x = "diabetesMed", y="readmitted",data = df, height = 6, kind ="bar")
g.set_ylabels("Probability of Readmitted")
plt.show()
```



In [151…

```
df.isnull().sum()
```

Out[151…

```
encounter_id                  0
patient_nbr                   0
race                          0
gender                        0
age                           0
admission_type_id         10396
discharge_disposition_id   4680
admission_source_id        6942
time_in_hospital              0
num_lab_procedures            0
num_procedures                0
num_medications               0
number_outpatient             0
number_emergency              0
number_inpatient              0
diag_1                        0
diag_2                        0
diag_3                        0
number_diagnoses              0
max_glu_serum                 0
A1Cresult                     0
metformin                     0
repaglinide                   0
nateglinide                   0
chlorpropamide                0
glimepiride                   0
acetohexamide                 0
glipizide                     0
glyburide                     0
tolbutamide                   0
pioglitazone                  0
```

```
rosiglitazone                 0
acarbose                      0
miglitol                      0
troglitazone                  0
tolazamide                    0
insulin                       0
glyburide-metformin           0
glipizide-metformin           0
glimepiride-pioglitazone      0
metformin-rosiglitazone       0
metformin-pioglitazone        0
change                        0
diabetesMed                   0
readmitted                    0
dtype: int64
```

In [152…
```
df['race'] = df['race'].fillna(df['race'].mode()[0])
```

In [153…
```
df['admission_type_id'] = df['admission_type_id'].fillna(df['admission_type_id'].mode()
```

In [154…
```
df['discharge_disposition_id'] = df['discharge_disposition_id'].fillna(df['discharge_di
```

In [155…
```
df['admission_source_id'] = df['admission_source_id'].fillna(df['admission_source_id'].
```

In [156…
```
df.head()
```

Out[156…

	encounter_id	patient_nbr	race	gender	age	admission_type_id	discharge_disposition_id
0	2278392	8222157	Caucasian	Female	5	Emergency	Discharged to Home
1	149190	55629189	Caucasian	Female	15	Emergency	Discharged to Home
2	64410	86047875	AfricanAmerican	Female	25	Emergency	Discharged to Home
3	500364	82442376	Caucasian	Male	35	Emergency	Discharged to Home
4	16680	42519267	Caucasian	Male	45	Emergency	Discharged to Home

5 rows × 45 columns

CODE

In [157…
```
df.shape
```

Out[157…
```
(101763, 45)
```

In [158…
```
df.head()
```

Out[158...

	encounter_id	patient_nbr	race	gender	age	admission_type_id	discharge_disposition_id
0	2278392	8222157	Caucasian	Female	5	Emergency	Discharged to Home
1	149190	55629189	Caucasian	Female	15	Emergency	Discharged to Home
2	64410	86047875	AfricanAmerican	Female	25	Emergency	Discharged to Home
3	500364	82442376	Caucasian	Male	35	Emergency	Discharged to Home
4	16680	42519267	Caucasian	Male	45	Emergency	Discharged to Home

5 rows × 45 columns

In [159...
```
df.drop(['encounter_id','patient_nbr'],axis=1,inplace=True)
```

In [160...
```
df.shape
```

Out[160... (101763, 43)

Dropping the duplicates entries

In [161...
```
cat_data = df.select_dtypes('O')

num_data = df.select_dtypes(np.number)

cat_data
```

Out[161...

	race	gender	admission_type_id	discharge_disposition_id	admission_source_id	di
0	Caucasian	Female	Emergency	Discharged to Home	Referral	Dia
1	Caucasian	Female	Emergency	Discharged to Home	Emergency	(
2	AfricanAmerican	Female	Emergency	Discharged to Home	Emergency	(
3	Caucasian	Male	Emergency	Discharged to Home	Emergency	(
4	Caucasian	Male	Emergency	Discharged to Home	Emergency	Neopl
...	...	...	...	...	...	
101761	AfricanAmerican	Male	Emergency	Other	Emergency	Dia
101762	AfricanAmerican	Female	Emergency	Other	Other	Dige
101763	Caucasian	Male	Emergency	Discharged to Home	Emergency	(
101764	Caucasian	Female	Emergency	Other	Emergency	I
101765	Caucasian	Male	Emergency	Discharged to Home	Emergency	Dige

101763 rows × 31 columns

In [162…
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

for i in cat_data:
    cat_data[i] = le.fit_transform(cat_data[i])
```

```
<ipython-input-162-74041ae2a149>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  cat_data[i] = le.fit_transform(cat_data[i])
```

In [163…
```
df = pd.concat([num_data,cat_data],axis=1)
df.head()
```

Out[163…

	age	time_in_hospital	num_lab_procedures	num_procedures	num_medications	number_outpatient	n
0	5	1	41	0	1	0	
1	15	3	59	0	18	0	
2	25	2	11	5	13	2	
3	35	2	44	1	16	0	
4	45	1	51	0	8	0	

5 rows × 43 columns

Splitting the dependent and independent variable¶

In [164…
```
X = df.drop('readmitted',axis=1)

y = df['readmitted']
```

In [165…
```
X.shape, y.shape
```

Out[165… ((101763, 42), (101763,))

In [166…
```
#modelling now
import statsmodels.api as sm
from sklearn.preprocessing import StandardScaler
```

In [167…
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=99)
```

In [168…
```
SC = StandardScaler()

X_train_scaled = pd.DataFrame(SC.fit_transform(X_train),columns=X_train.columns)
X_test_scaled = pd.DataFrame(SC.transform(X_test),columns=X_test.columns)
```

In [169…
```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[169… ((71234, 42), (30529, 42), (71234,), (30529,))

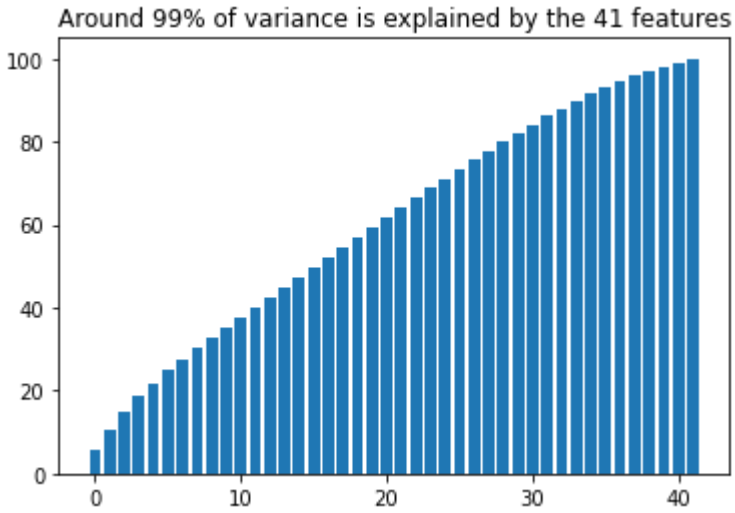
PCA

In [170…
```
from sklearn.decomposition import PCA
pca = PCA()
X_train_pca=pca.fit_transform(X_train_scaled)
```

In [171…
```
total=sum(pca.explained_variance_)
k=0
current_variance=0
while current_variance/total < 0.99:
    current_variance += pca.explained_variance_[k]
    k=k+1
k
```

Out[171… 41

In [172…
```
import matplotlib.pyplot as plt
cum_sum = pca.explained_variance_ratio_.cumsum()
cum_sum = cum_sum*100
plt.bar(range(k+1), cum_sum)
plt.title("Around 99% of variance is explained by the {} features".format(k));
```



In [173…

```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
X_pca = sc.fit_transform(X_train_pca)
```

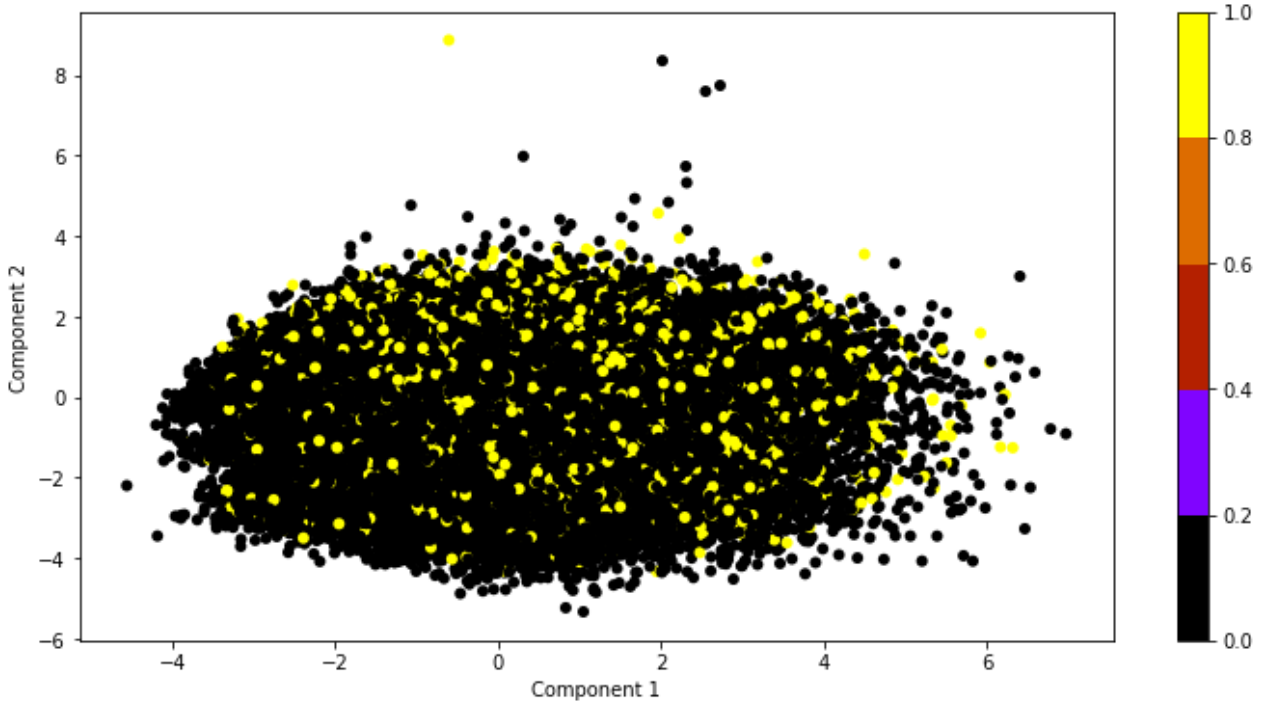
In [174…

```
pca = PCA(2)
projected = pca.fit_transform(X_train_pca)
print(X_train_pca.shape)
print(projected.shape)
```

```
(71234, 42)
(71234, 2)
```

In [175…

```
plt.figure(figsize=(12, 6))
plt.scatter(projected[:, 0],
projected[:, 1],
c=y_train,
edgecolor='none',
alpha=1,
cmap=plt.cm.get_cmap('gnuplot', 5)
)
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.colorbar();
```



MODELS

LOGISTIC REGRESSION

In [176…

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(fit_intercept=True, penalty='l2')
```

```
LR.fit(X_train_scaled,y_train)
#LR.fit(X_train,y_train)
```

Out[176… LogisticRegression()

In [177…
```
y_pred_LR = LR.predict(X_test_scaled)
#y_pred_LR = LR.predict(X_test)
```

In [178…
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred_LR)
```

Out[178…
```
array([[27089,    54],
       [ 3340,    46]], dtype=int64)
```

In [179…
```
from sklearn.metrics import classification_report
print("Classification report - \n", classification_report(y_test,y_pred_LR))
```

```
Classification report -
               precision    recall  f1-score   support

           0       0.89      1.00      0.94     27143
           1       0.46      0.01      0.03      3386

    accuracy                           0.89     30529
   macro avg       0.68      0.51      0.48     30529
weighted avg       0.84      0.89      0.84     30529
```

In [180…
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred_LR)
```

Out[180… 0.8888270169347178

SMOTE LR

In [181…
```
X.shape, y.shape
```

Out[181… ((101763, 42), (101763,))

In [182…
```
print('Original dataset shape {}'.format(Counter(y)))
smt = SMOTE(random_state=20)
train_input_new, train_output_new = smt.fit_resample(X, y)
print('New dataset shape {}'.format(Counter(train_output_new)))
train_input_new = pd.DataFrame(train_input_new, columns = list(X.columns))
X_train, X_test, y_train, y_test = train_test_split(train_input_new, train_output_new,
```

```
Original dataset shape Counter({0: 90406, 1: 11357})
New dataset shape Counter({0: 90406, 1: 90406})
```

In [184…
```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(fit_intercept=True, penalty='l2')
```

```
LR.fit(X_train,y_train)
#LR.fit(X_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:763: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[184… LogisticRegression()

In [185…
```
y_pred_LR = LR.predict(X_test)
#y_pred_LR = LR.predict(X_test)
```

In [186…
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred_LR)
```

Out[186…
```
array([[12437,  5487],
       [ 4977, 13262]], dtype=int64)
```

In [187…
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred_LR)
```

Out[187… 0.7106434753753836

RANDOM FOREST

In [188…
```
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier()

#RF.fit(X_train_scaled,y_train)
RF.fit(X_train,y_train)
```

Out[188… RandomForestClassifier()

In [189…
```
#y_pred_RF = RF.predict(X_test_scaled)
y_pred_RF = RF.predict(X_test)
```

In [190…
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred_RF)
```

Out[190…
```
array([[16159,  1765],
       [ 2415, 15824]], dtype=int64)
```

In [191…
```
print("Classification report - \n", classification_report(y_test,y_pred_RF))
```

```
Classification report -
```

```
                precision    recall  f1-score   support

           0       0.87      0.90      0.89     17924
           1       0.90      0.87      0.88     18239

    accuracy                           0.88     36163
   macro avg       0.88      0.88      0.88     36163
weighted avg       0.88      0.88      0.88     36163
```

In [192…
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred_RF)
```

Out[192… 0.8844122445593563

plt.figure(figsize=(12,24)) sns.barplot(sorted(RF.feature*importances*),X_train.columns);

z = pd.DataFrame([RF.feature*importances*,X_train.columns]).T

z.columns = ['Feature','importance'] plt.figure(figsize=(13,12))

sns.barplot(y=z['Feature'],x=z['importance'])

important_features = pd.DataFrame({'Features': X_train.columns,'Importance': RF.feature*importances*}) important_features = important_features.sort_values('Importance', ascending = False) plt.figure(figsize=(12,24)) sns.barplot(x = 'Importance', y = 'Features', data = important_features) plt.title('Feature Importance', fontsize = 15) plt.xlabel('Importance', fontsize = 15) plt.ylabel('Features', fontsize = 15) plt.show()

Decision Tree

In [193…
```
from sklearn.tree import DecisionTreeClassifier#for checking testing results
from sklearn.metrics import classification_report, confusion_matrix#for visualizing tre
```

In [194…
```
# Defining the decision tree algorithm
dtree=DecisionTreeClassifier()
dtree.fit(X_train,y_train)
print('Decision Tree Classifier Created')
```

```
Decision Tree Classifier Created
```

In [195…
```
# Predicting the values of test data
y_pred_DT = dtree.predict(X_test)
print("Classification report - \n", classification_report(y_test,y_pred_DT))
```

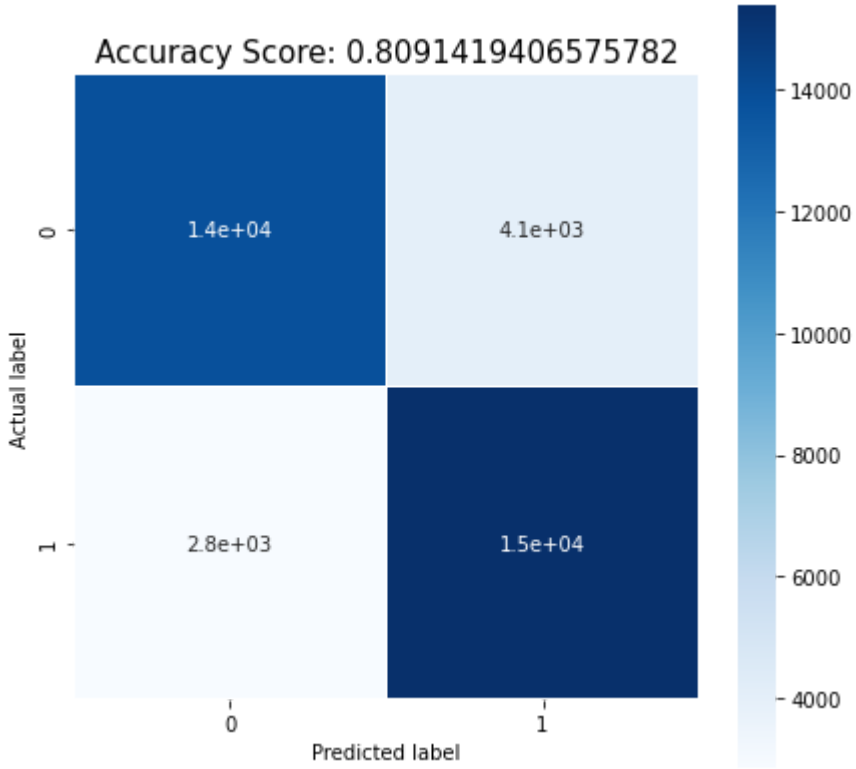
```
Classification report -
                precision    recall  f1-score   support

           0       0.83      0.77      0.80     17924
           1       0.79      0.84      0.82     18239

    accuracy                           0.81     36163
   macro avg       0.81      0.81      0.81     36163
weighted avg       0.81      0.81      0.81     36163
```

In [196…
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred_DT)
```

Out[196… 0.8091419406575782

In [197…
```
cm = confusion_matrix(y_test, y_pred_DT)
plt.figure(figsize=(7,7))
sns.heatmap(data=cm,linewidths=.5, annot=True,square = True,  cmap = 'Blues')
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
all_sample_title = 'Accuracy Score: {0}'.format(dtree.score(X_test, y_test))
plt.title(all_sample_title, size = 15)
```

Out[197… Text(0.5, 1.0, 'Accuracy Score: 0.8091419406575782')



SVM

from sklearn.svm import SVC classifier = SVC(kernel='rbf', random_state = 1)

classifier.fit(X_train,y_train)

Y_pred_SVM = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix cm = confusion_matrix(y_test,Y_pred_SVM) cm

from sklearn.metrics import accuracy_score accuracy_score(y_test,Y_pred_SVM)

Neural Network

In [198…
```
from keras.models import Sequential
```

```
from keras.layers import Dense
from keras.layers import Dropout
```

In [201…

```
# define the keras model
model = Sequential()
model.add(Dense(32, input_dim=42, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(8, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=300, batch_size=1000)
_, accuracy = model.evaluate(X_test, y_test)
print('Accuracy: %.2f' % (accuracy*100))
```

```
Epoch 1/300
145/145 [==============================] - 1s 3ms/step - loss: 0.9628 - accuracy: 0.4989
Epoch 2/300
145/145 [==============================] - 0s 3ms/step - loss: 0.7055 - accuracy: 0.5016
Epoch 3/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6979 - accuracy: 0.4994
Epoch 4/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6950 - accuracy: 0.5031
Epoch 5/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6947 - accuracy: 0.5065
Epoch 6/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6936 - accuracy: 0.5096
Epoch 7/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6921 - accuracy: 0.5187
Epoch 8/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6848 - accuracy: 0.5467
Epoch 9/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6662 - accuracy: 0.5899
Epoch 10/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6548 - accuracy: 0.6125
Epoch 11/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6497 - accuracy: 0.6212
Epoch 12/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6477 - accuracy: 0.6253
Epoch 13/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6458 - accuracy: 0.6275
Epoch 14/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6433 - accuracy: 0.6323
Epoch 15/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6425 - accuracy: 0.6337
Epoch 16/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6408 - accuracy: 0.6363
Epoch 17/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6394 - accuracy: 0.6378
Epoch 18/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6389 - accuracy: 0.6385
Epoch 19/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6385 - accuracy: 0.6390
Epoch 20/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6373 - accuracy: 0.6414
Epoch 21/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6363 - accuracy: 0.6427
Epoch 22/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6353 - accuracy: 0.6438
Epoch 23/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6351 - accuracy: 0.6437
Epoch 24/300
```

```
145/145 [==============================] - 0s 3ms/step - loss: 0.6315 - accuracy: 0.6447
Epoch 25/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6212 - accuracy: 0.6432
Epoch 26/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6122 - accuracy: 0.6479
Epoch 27/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6096 - accuracy: 0.6512
Epoch 28/300
145/145 [==============================] - 0s 3ms/step - loss: 0.6017 - accuracy: 0.6681
Epoch 29/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5955 - accuracy: 0.6863
Epoch 30/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5936 - accuracy: 0.6925
Epoch 31/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5910 - accuracy: 0.6928
Epoch 32/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5894 - accuracy: 0.6949
Epoch 33/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5870 - accuracy: 0.6959
Epoch 34/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5867 - accuracy: 0.6957
Epoch 35/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5823 - accuracy: 0.6982
Epoch 36/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5792 - accuracy: 0.6983
Epoch 37/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5763 - accuracy: 0.6995
Epoch 38/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5748 - accuracy: 0.6991
Epoch 39/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5717 - accuracy: 0.7008
Epoch 40/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5704 - accuracy: 0.7023
Epoch 41/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5678 - accuracy: 0.7012
Epoch 42/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5614 - accuracy: 0.7120
Epoch 43/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5560 - accuracy: 0.7237
Epoch 44/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5540 - accuracy: 0.7237
Epoch 45/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5516 - accuracy: 0.7241
Epoch 46/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5488 - accuracy: 0.7254
Epoch 47/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5477 - accuracy: 0.7269
Epoch 48/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5451 - accuracy: 0.7289
Epoch 49/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5435 - accuracy: 0.7303
Epoch 50/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5439 - accuracy: 0.7288
Epoch 51/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5417 - accuracy: 0.7321
Epoch 52/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5415 - accuracy: 0.7303
Epoch 53/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5405 - accuracy: 0.7333
Epoch 54/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5379 - accuracy: 0.7340
Epoch 55/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5372 - accuracy: 0.7348
Epoch 56/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5357 - accuracy: 0.7347
```

```
Epoch 57/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5350 - accuracy: 0.7353
Epoch 58/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5353 - accuracy: 0.7349
Epoch 59/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5326 - accuracy: 0.7365
Epoch 60/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5323 - accuracy: 0.7368
Epoch 61/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5313 - accuracy: 0.7370
Epoch 62/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5301 - accuracy: 0.7381
Epoch 63/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5291 - accuracy: 0.7391
Epoch 64/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5285 - accuracy: 0.7385
Epoch 65/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5290 - accuracy: 0.7397
Epoch 66/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5271 - accuracy: 0.7403
Epoch 67/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5290 - accuracy: 0.7373
Epoch 68/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5262 - accuracy: 0.7399
Epoch 69/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5247 - accuracy: 0.7397
Epoch 70/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5235 - accuracy: 0.7420
Epoch 71/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5248 - accuracy: 0.7416
Epoch 72/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5233 - accuracy: 0.7430
Epoch 73/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5218 - accuracy: 0.7426
Epoch 74/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5234 - accuracy: 0.7413
Epoch 75/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5219 - accuracy: 0.7419
Epoch 76/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5198 - accuracy: 0.7443
Epoch 77/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5206 - accuracy: 0.7458
Epoch 78/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5208 - accuracy: 0.7445
Epoch 79/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5195 - accuracy: 0.7440
Epoch 80/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5195 - accuracy: 0.7446
Epoch 81/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5183 - accuracy: 0.7450
Epoch 82/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5195 - accuracy: 0.7446
Epoch 83/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5171 - accuracy: 0.7461
Epoch 84/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5180 - accuracy: 0.7478
Epoch 85/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5183 - accuracy: 0.7454
Epoch 86/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5162 - accuracy: 0.7474
Epoch 87/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5155 - accuracy: 0.7469
Epoch 88/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5159 - accuracy: 0.7465
Epoch 89/300
```

```
145/145 [==============================] - 0s 3ms/step - loss: 0.5155 - accuracy: 0.7468
Epoch 90/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5161 - accuracy: 0.7469
Epoch 91/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5154 - accuracy: 0.7466
Epoch 92/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5159 - accuracy: 0.7471
Epoch 93/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5145 - accuracy: 0.7472
Epoch 94/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5143 - accuracy: 0.7481
Epoch 95/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5141 - accuracy: 0.7464
Epoch 96/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5141 - accuracy: 0.7474
Epoch 97/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5129 - accuracy: 0.7484
Epoch 98/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5131 - accuracy: 0.7465
Epoch 99/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5120 - accuracy: 0.7486
Epoch 100/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5120 - accuracy: 0.7484
Epoch 101/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5118 - accuracy: 0.7489
Epoch 102/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5112 - accuracy: 0.7483
Epoch 103/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5108 - accuracy: 0.7484
Epoch 104/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5097 - accuracy: 0.7493
Epoch 105/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5095 - accuracy: 0.7499
Epoch 106/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5082 - accuracy: 0.7498
Epoch 107/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5097 - accuracy: 0.7496
Epoch 108/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5096 - accuracy: 0.7502
Epoch 109/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5083 - accuracy: 0.7497
Epoch 110/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5086 - accuracy: 0.7501
Epoch 111/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5074 - accuracy: 0.7497
Epoch 112/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5090 - accuracy: 0.7503
Epoch 113/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5083 - accuracy: 0.7498
Epoch 114/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5082 - accuracy: 0.7501
Epoch 115/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5077 - accuracy: 0.749
8: 0s - loss: 0.5095 - accura
Epoch 116/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5074 - accuracy: 0.7497
Epoch 117/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5056 - accuracy: 0.7523
Epoch 118/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5070 - accuracy: 0.7512
Epoch 119/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5068 - accuracy: 0.7511
Epoch 120/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5059 - accuracy: 0.7515
Epoch 121/300
```

```
145/145 [==============================] - 0s 3ms/step - loss: 0.5063 - accuracy: 0.7509
Epoch 122/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5059 - accuracy: 0.7513
Epoch 123/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5060 - accuracy: 0.7511
Epoch 124/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5057 - accuracy: 0.7511
Epoch 125/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5068 - accuracy: 0.7518
Epoch 126/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5048 - accuracy: 0.7518
Epoch 127/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5068 - accuracy: 0.7511
Epoch 128/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5045 - accuracy: 0.7522
Epoch 129/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5051 - accuracy: 0.7516
Epoch 130/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5046 - accuracy: 0.7529
Epoch 131/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5032 - accuracy: 0.7529
Epoch 132/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5041 - accuracy: 0.7518
Epoch 133/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5042 - accuracy: 0.7526
Epoch 134/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5040 - accuracy: 0.7525
Epoch 135/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5049 - accuracy: 0.7531
Epoch 136/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5050 - accuracy: 0.7523
Epoch 137/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5029 - accuracy: 0.7529
Epoch 138/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5039 - accuracy: 0.7518
Epoch 139/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5031 - accuracy: 0.7535
Epoch 140/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5034 - accuracy: 0.7538
Epoch 141/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5027 - accuracy: 0.7520
Epoch 142/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5049 - accuracy: 0.7519
Epoch 143/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5036 - accuracy: 0.7534
Epoch 144/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5057 - accuracy: 0.7516
Epoch 145/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5027 - accuracy: 0.7527
Epoch 146/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5028 - accuracy: 0.7528
Epoch 147/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5032 - accuracy: 0.7534
Epoch 148/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5036 - accuracy: 0.7534
Epoch 149/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5025 - accuracy: 0.7536
Epoch 150/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5016 - accuracy: 0.7537
Epoch 151/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5015 - accuracy: 0.7534
Epoch 152/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5014 - accuracy: 0.7536
Epoch 153/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4999 - accuracy: 0.7542
```

```
Epoch 154/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5047 - accuracy: 0.7528
Epoch 155/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5027 - accuracy: 0.7535
Epoch 156/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5021 - accuracy: 0.7552
Epoch 157/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5039 - accuracy: 0.7535
Epoch 158/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5005 - accuracy: 0.7557
Epoch 159/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5022 - accuracy: 0.7535
Epoch 160/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5017 - accuracy: 0.7548
Epoch 161/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5005 - accuracy: 0.7543
Epoch 162/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5034 - accuracy: 0.7535
Epoch 163/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5031 - accuracy: 0.7531
Epoch 164/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5007 - accuracy: 0.7547
Epoch 165/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5009 - accuracy: 0.7549
Epoch 166/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5010 - accuracy: 0.7530
Epoch 167/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5019 - accuracy: 0.7529
Epoch 168/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5009 - accuracy: 0.7546
Epoch 169/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5006 - accuracy: 0.7545
Epoch 170/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5012 - accuracy: 0.7537
Epoch 171/300
145/145 [==============================] - 0s 2ms/step - loss: 0.4999 - accuracy: 0.7552
Epoch 172/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5008 - accuracy: 0.7556
Epoch 173/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5005 - accuracy: 0.7537
Epoch 174/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5007 - accuracy: 0.7545
Epoch 175/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5007 - accuracy: 0.7539
Epoch 176/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5002 - accuracy: 0.7551
Epoch 177/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5013 - accuracy: 0.7549
Epoch 178/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5002 - accuracy: 0.7560
Epoch 179/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4996 - accuracy: 0.7565
Epoch 180/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4997 - accuracy: 0.7541
Epoch 181/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4997 - accuracy: 0.7540
Epoch 182/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4996 - accuracy: 0.7548
Epoch 183/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5003 - accuracy: 0.7558
Epoch 184/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4993 - accuracy: 0.7548
Epoch 185/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5006 - accuracy: 0.7552
Epoch 186/300
```

```
145/145 [==============================] - 0s 3ms/step - loss: 0.5023 - accuracy: 0.7550
Epoch 187/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5016 - accuracy: 0.7539
Epoch 188/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4998 - accuracy: 0.7548
Epoch 189/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4996 - accuracy: 0.7537
Epoch 190/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4988 - accuracy: 0.7550
Epoch 191/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4988 - accuracy: 0.7547
Epoch 192/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4987 - accuracy: 0.7561
Epoch 193/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5001 - accuracy: 0.7548
Epoch 194/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4987 - accuracy: 0.7558
Epoch 195/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4976 - accuracy: 0.7563
Epoch 196/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4977 - accuracy: 0.7566
Epoch 197/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4979 - accuracy: 0.7558
Epoch 198/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4987 - accuracy: 0.755
8: 0s - loss: 0.4970 - accu
Epoch 199/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4987 - accuracy: 0.7560
Epoch 200/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5037 - accuracy: 0.7529
Epoch 201/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4980 - accuracy: 0.7556
Epoch 202/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4976 - accuracy: 0.7554
Epoch 203/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4966 - accuracy: 0.7567
Epoch 204/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4984 - accuracy: 0.7550
Epoch 205/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4991 - accuracy: 0.7556
Epoch 206/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4980 - accuracy: 0.7568
Epoch 207/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4982 - accuracy: 0.7548
Epoch 208/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4996 - accuracy: 0.7543
Epoch 209/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4983 - accuracy: 0.7564
Epoch 210/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4965 - accuracy: 0.7576
Epoch 211/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4962 - accuracy: 0.7562
Epoch 212/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4982 - accuracy: 0.7568
Epoch 213/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4963 - accuracy: 0.7575
Epoch 214/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4981 - accuracy: 0.7559
Epoch 215/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4972 - accuracy: 0.7575
Epoch 216/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4983 - accuracy: 0.7550
Epoch 217/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4972 - accuracy: 0.7565
Epoch 218/300
```

```
145/145 [==============================] - 0s 3ms/step - loss: 0.4985 - accuracy: 0.7564
Epoch 219/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4962 - accuracy: 0.7570
Epoch 220/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4953 - accuracy: 0.7567
Epoch 221/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4971 - accuracy: 0.7565
Epoch 222/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4964 - accuracy: 0.7570
Epoch 223/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4960 - accuracy: 0.7563
Epoch 224/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4961 - accuracy: 0.7566
Epoch 225/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4944 - accuracy: 0.7582
Epoch 226/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4963 - accuracy: 0.7565
Epoch 227/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4952 - accuracy: 0.7574
Epoch 228/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4960 - accuracy: 0.7565
Epoch 229/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4962 - accuracy: 0.7571
Epoch 230/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4972 - accuracy: 0.7556
Epoch 231/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4948 - accuracy: 0.7579
Epoch 232/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4949 - accuracy: 0.7563
Epoch 233/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4958 - accuracy: 0.7581
Epoch 234/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4987 - accuracy: 0.7563
Epoch 235/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4942 - accuracy: 0.7572
Epoch 236/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4954 - accuracy: 0.7562
Epoch 237/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4941 - accuracy: 0.7568
Epoch 238/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4955 - accuracy: 0.7574
Epoch 239/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4943 - accuracy: 0.7570
Epoch 240/300
145/145 [==============================] - 0s 2ms/step - loss: 0.4944 - accuracy: 0.7577
Epoch 241/300
145/145 [==============================] - 0s 2ms/step - loss: 0.4946 - accuracy: 0.7580
Epoch 242/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4983 - accuracy: 0.7556
Epoch 243/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4938 - accuracy: 0.7564
Epoch 244/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4936 - accuracy: 0.7577
Epoch 245/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4948 - accuracy: 0.7580
Epoch 246/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4956 - accuracy: 0.7553
Epoch 247/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4932 - accuracy: 0.7586
Epoch 248/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4952 - accuracy: 0.7566
Epoch 249/300
145/145 [==============================] - 0s 2ms/step - loss: 0.4938 - accuracy: 0.7576
Epoch 250/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4954 - accuracy: 0.7565
```

```
Epoch 251/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4929 - accuracy: 0.7577
Epoch 252/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4937 - accuracy: 0.7583
Epoch 253/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4937 - accuracy: 0.7577
Epoch 254/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4942 - accuracy: 0.7566
Epoch 255/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4986 - accuracy: 0.7553
Epoch 256/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4927 - accuracy: 0.7585
Epoch 257/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4927 - accuracy: 0.7588
Epoch 258/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4934 - accuracy: 0.7570
Epoch 259/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4930 - accuracy: 0.7574
Epoch 260/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4926 - accuracy: 0.7579
Epoch 261/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4912 - accuracy: 0.7588
Epoch 262/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4929 - accuracy: 0.7584
Epoch 263/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4922 - accuracy: 0.7582
Epoch 264/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4931 - accuracy: 0.7576
Epoch 265/300
145/145 [==============================] - 0s 3ms/step - loss: 0.4925 - accuracy: 0.7582
Epoch 266/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5029 - accuracy: 0.7537
Epoch 267/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5047 - accuracy: 0.7518
Epoch 268/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5040 - accuracy: 0.7519
Epoch 269/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5039 - accuracy: 0.7521
Epoch 270/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5041 - accuracy: 0.7516
Epoch 271/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5030 - accuracy: 0.752
9: 0s - loss: 0.5027 - accuracy: 0.
Epoch 272/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5026 - accuracy: 0.7528
Epoch 273/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5028 - accuracy: 0.7520
Epoch 274/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5019 - accuracy: 0.7534
Epoch 275/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5022 - accuracy: 0.7529
Epoch 276/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5027 - accuracy: 0.7543
Epoch 277/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5027 - accuracy: 0.7543
Epoch 278/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5031 - accuracy: 0.7527
Epoch 279/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5026 - accuracy: 0.7530
Epoch 280/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5027 - accuracy: 0.7533
Epoch 281/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5014 - accuracy: 0.7548
Epoch 282/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5012 - accuracy: 0.753
```

```
8: 0s - loss: 0.5011 - accuracy: 0.75
Epoch 283/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5035 - accuracy: 0.7539
Epoch 284/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5021 - accuracy: 0.7533
Epoch 285/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5023 - accuracy: 0.7530
Epoch 286/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5027 - accuracy: 0.7525
Epoch 287/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5019 - accuracy: 0.7535
Epoch 288/300
145/145 [==============================] - 0s 2ms/step - loss: 0.5020 - accuracy: 0.7542
Epoch 289/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5009 - accuracy: 0.7546
Epoch 290/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5016 - accuracy: 0.7543
Epoch 291/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5014 - accuracy: 0.7547
Epoch 292/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5055 - accuracy: 0.7517
Epoch 293/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5034 - accuracy: 0.7542
Epoch 294/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5015 - accuracy: 0.7525
Epoch 295/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5012 - accuracy: 0.7546
Epoch 296/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5208 - accuracy: 0.7477
Epoch 297/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5282 - accuracy: 0.7449
Epoch 298/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5257 - accuracy: 0.7484
Epoch 299/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5256 - accuracy: 0.7480
Epoch 300/300
145/145 [==============================] - 0s 3ms/step - loss: 0.5256 - accuracy: 0.7475
1131/1131 [==============================] - 1s 917us/step - loss: 0.4984 - accuracy: 0.
75970s - loss: 0.4
Accuracy: 75.97
```

In [ ]: