Loading, Cross-Checking and Cleaning of the data

Python provides tools that assist with data analysis and manipulation In this section pandas will be used.

```
# import libraries
import pandas as pd
```

```
df_meta = pd.read_excel('PS206767-553247439.xls', sheet_name='Metadata')
df_meta.head()
```

	Phenotype Data File Name	Warfarin Consortium Combined Data Set, March 2008	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	NaN	NaN	NaN	NaN	NaN
1	Group	IWPC	NaN	NaN	NaN
2	Investigator	International Warfarin Pharmacogenetics Consor...	NaN	NaN	NaN
3	Contact	International Warfarin Pharmacogenetics Consor...	NaN	NaN	NaN
4	Reference	see http://www.pharmgkb.org/views/project.jsp?...	NaN	NaN	NaN

```
# data is in second worksheet
df = pd.read_excel('PS206767-553247439.xls', sheet_name='Subject Data')
df.head()
```

	PharmGKB Subject ID	PharmGKB Sample ID	Project Site	Gender	Race (Reported)	Race (OMB)	Ethnicity (Reported)	Ethnicity (OMB)	Age	Height (cm)	...	VKORC1 QC genotype: -4451 C>A (861); Chr16:31018002; rs17880887; A/C	CYP2C9 consensus	co
0	PA135312261	PA135312629	1	male	White	White	not Hispanic or Latino	not Hispanic or Latino	60 - 69	193.040	...	NaN	*1/*1	
1	PA135312262	PA135312630	1	female	White	White	not Hispanic or Latino	not Hispanic or Latino	50 - 59	176.530	...	C/C	*1/*1	
2	PA135312263	PA135312631	1	female	White	White	not Hispanic or Latino	not Hispanic or Latino	40 - 49	162.560	...	NaN	*1/*1	
3	PA135312264	PA135312632	1	male	White	White	not Hispanic or Latino	not Hispanic or Latino	60 - 69	182.245	...	NaN	*1/*1	
4	PA135312265	PA135312633	1	male	White	White	not Hispanic or Latino	not Hispanic or Latino	50 - 59	167.640	...	NaN	*1/*3	

5 rows × 68 columns

```
# load the dataset
data = df[["Gender","Race (Reported)","Age","Height (cm)","Weight (kg)","Diabetes","Simvastatin (Zocor)","Amiodarone (Cordarone)","Target IN
data
```

	Gender	Race (Reported)	Age	Height (cm)	Weight (kg)	Diabetes	Simvastatin (Zocor)	Amiodarone (Cordarone)	Target INR	INR on Reported Therapeutic Dose of Warfarin	Cyp2C9 genotypes	VKORC1 genotype: -1639 G>A (3673); chr16:31015190; rs9923231; C/T
0	male	White	60 - 69	193.040	115.70	NaN	0.0	0.0	2.5	2.60	*1/*1	A/G
1	female	White	50 - 59	176.530	144.20	NaN	0.0	0.0	2.5	2.15	*1/*1	A/A
2	female	White	40 - 49	162.560	77.10	NaN	0.0	0.0	2.5	1.90	*1/*1	G/G
3	male	White	60 - 69	182.245	90.70	NaN	0.0	0.0	2.5	2.40	*1/*1	A/G
4	male	White	50 - 59	167.640	72.60	NaN	0.0	0.0	2.5	1.90	*1/*3	A/G
...	...	...	...	...	...	...	...	...	...	...	...	...
5695	male	White	20 - 29	185.420	113.64	0.0	0.0	0.0	NaN	2.80	*1/*1	NaN
5696	female	White	70 - 79	160.020	55.91	0.0	0.0	0.0	NaN	2.80	*1/*3	NaN
			60									

```
# from .info() get data skeleton
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5700 entries, 0 to 5699
Data columns (total 13 columns):
 #   Column                                                             Non-Null Count  Dtype  
---  ------                                                             --------------  -----  
 0   Gender                                                             5696 non-null   object 
 1   Race (Reported)                                                    5194 non-null   object 
 2   Age                                                                5658 non-null   object 
 3   Height (cm)                                                        4554 non-null   float64
 4   Weight (kg)                                                        5413 non-null   float64
 5   Diabetes                                                           3283 non-null   float64
 6   Simvastatin (Zocor)                                                3861 non-null   float64
 7   Amiodarone (Cordarone)                                             4182 non-null   float64
 8   Target INR                                                         1259 non-null   float64
 9   INR on Reported Therapeutic Dose of Warfarin                       4968 non-null   float64
 10  Cyp2C9 genotypes                                                   5567 non-null   object 
 11  VKORC1 genotype: -1639 G>A (3673); chr16:31015190; rs9923231; C/T  4046 non-null   object 
 12  Therapeutic Dose of Warfarin                                       5528 non-null   float64
dtypes: float64(8), object(5)
memory usage: 579.0+ KB
```

```
# Count missing data from the dataset
data.isna().sum().sum()
```

```
14891
```

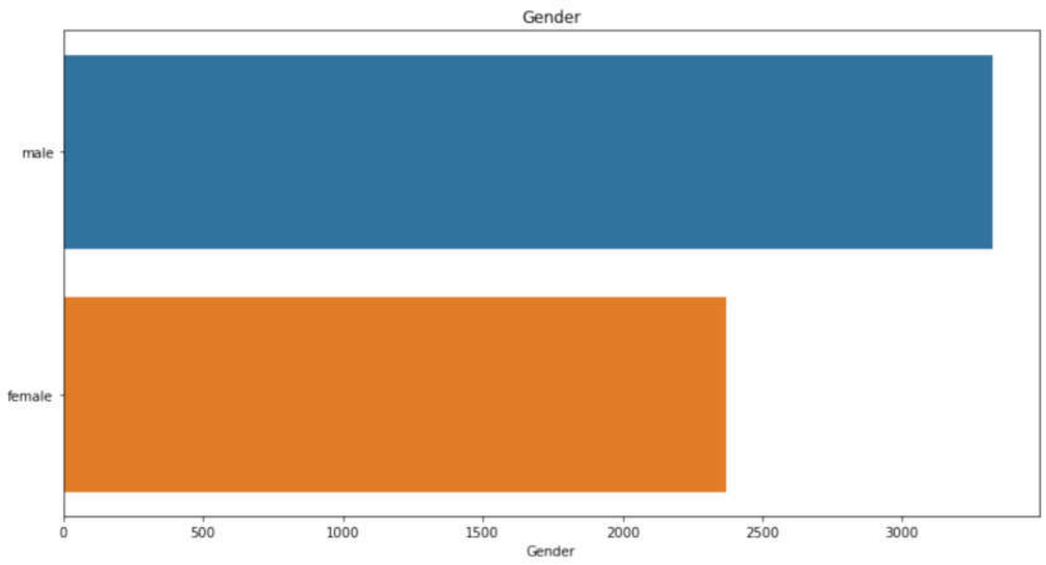
Data Visualization

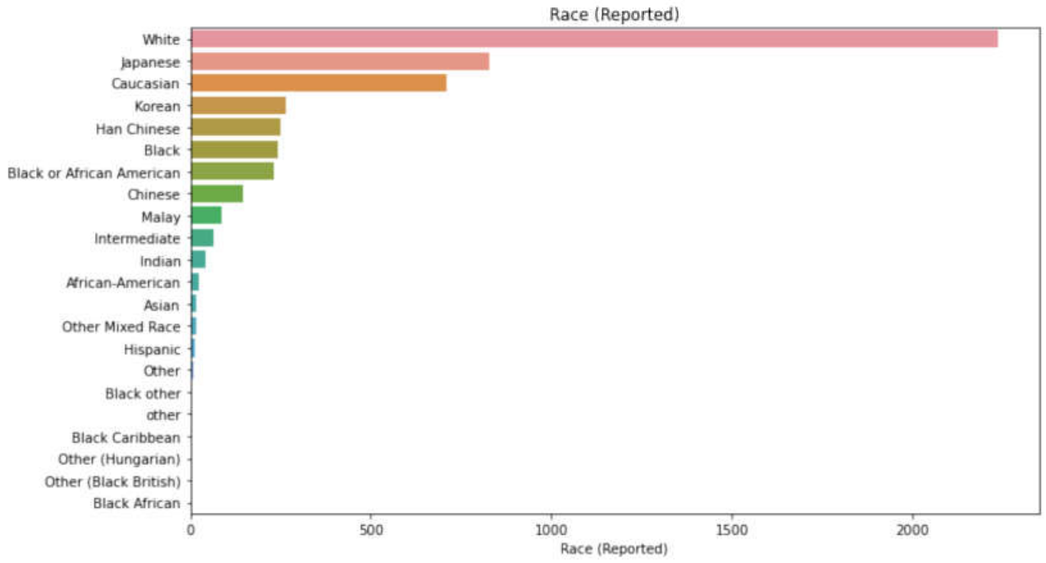
Import necessary tools to help visualize the data

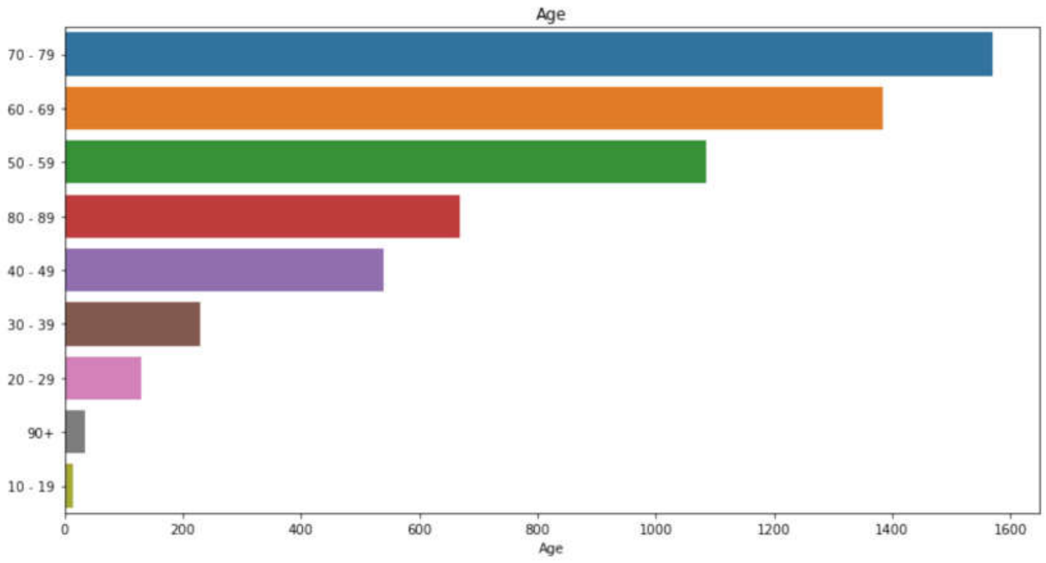
```
# import visulization libraries
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

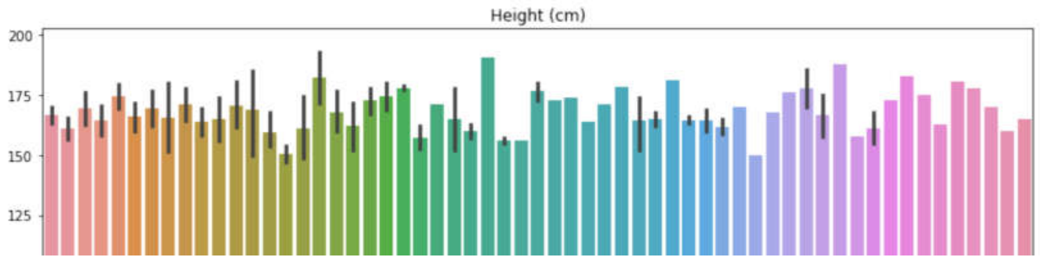
This is a descriptive analysis of how data is distributed in each column

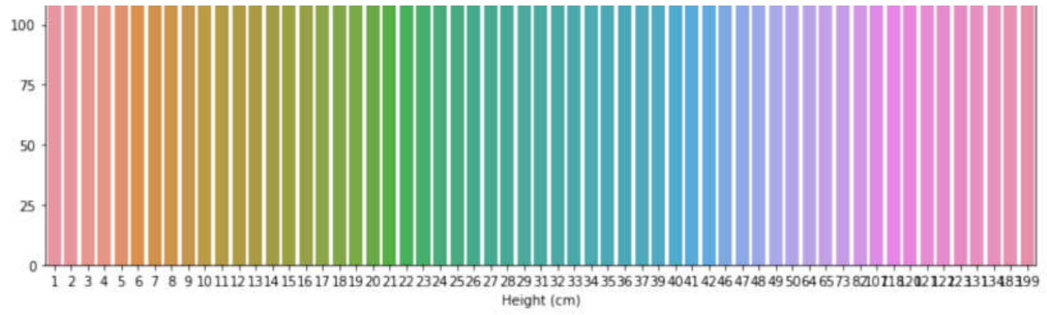
```
category = ["Gender","Race (Reported)","Age","Height (cm)","Weight (kg)","Diabetes","Simvastatin (Zocor)","Amiodarone (Cordarone)","Target I
for cat in category:
    plt.figure(figsize=(11,6))
    sns.barplot(data[cat].value_counts(),data[cat].value_counts().index,data=data)
    plt.title(cat)
    plt.tight_layout()
```

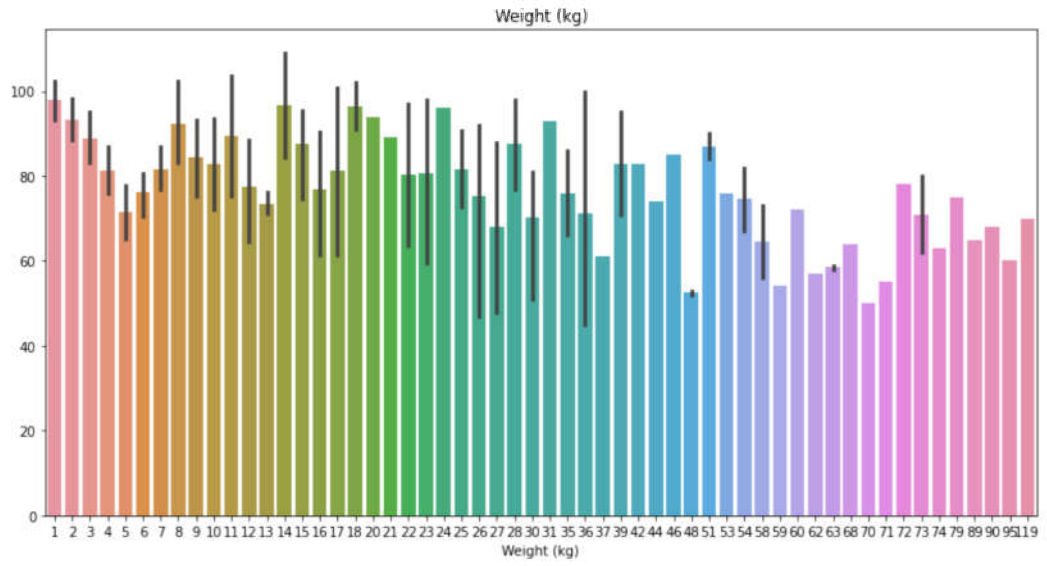

Gender
male
female
0
500
1000
1500
2000
2500
3000
Gender

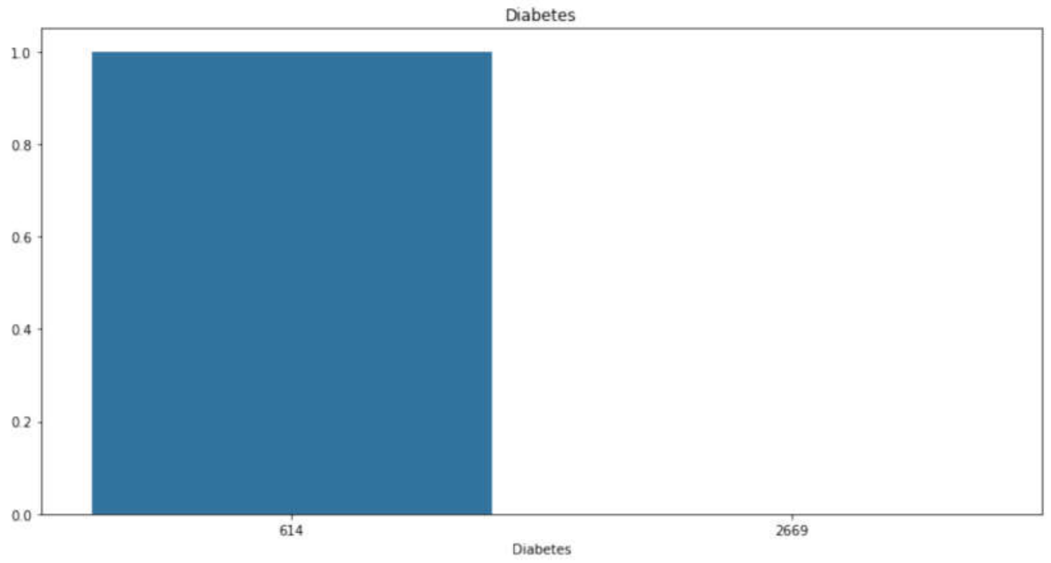

Race (Reported)
White
Japanese
Caucasian
Korean
Han Chinese
Black
Black or African American
Chinese
Malay
Intermediate
Indian
African-American
Asian
Other Mixed Race
Hispanic
Other
Black other
other
Black Caribbean
Other (Hungarian)
Other (Black British)
Black African
0
500
1000
1500
2000
Race (Reported)

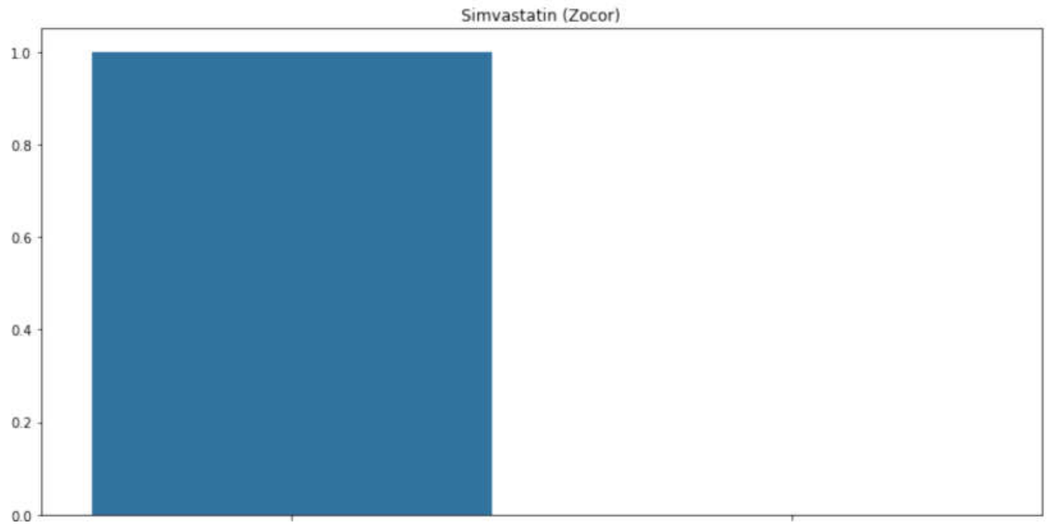

Age
70 - 79
60 - 69
50 - 59
80 - 89
40 - 49
30 - 39
20 - 29
90+
10 - 19
0
200
400
600
800
1000
1200
1400
1600
Age


Height (cm)
200
175
150
125

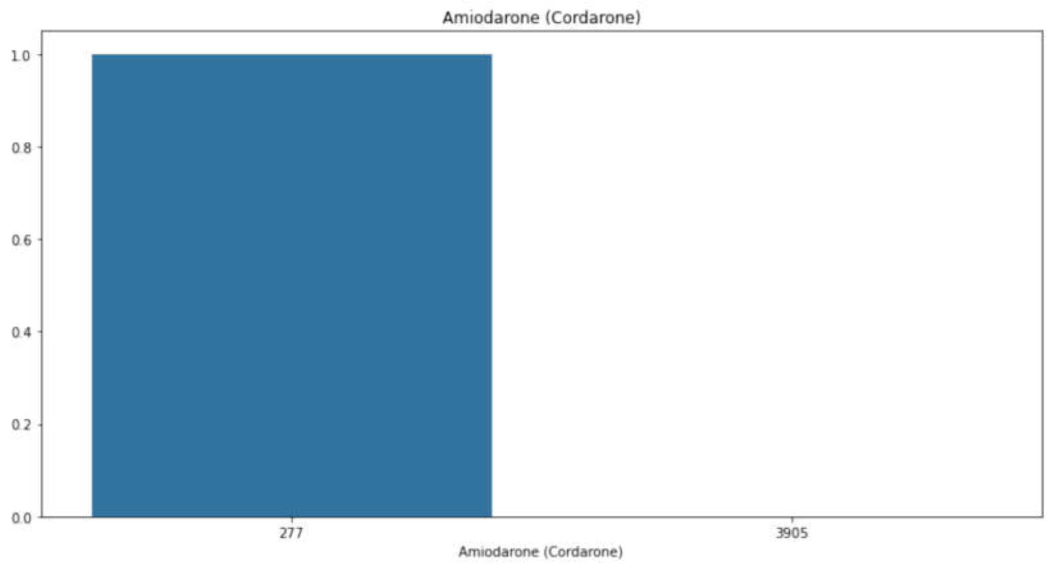

100
75
50
25
0
Height (cm)

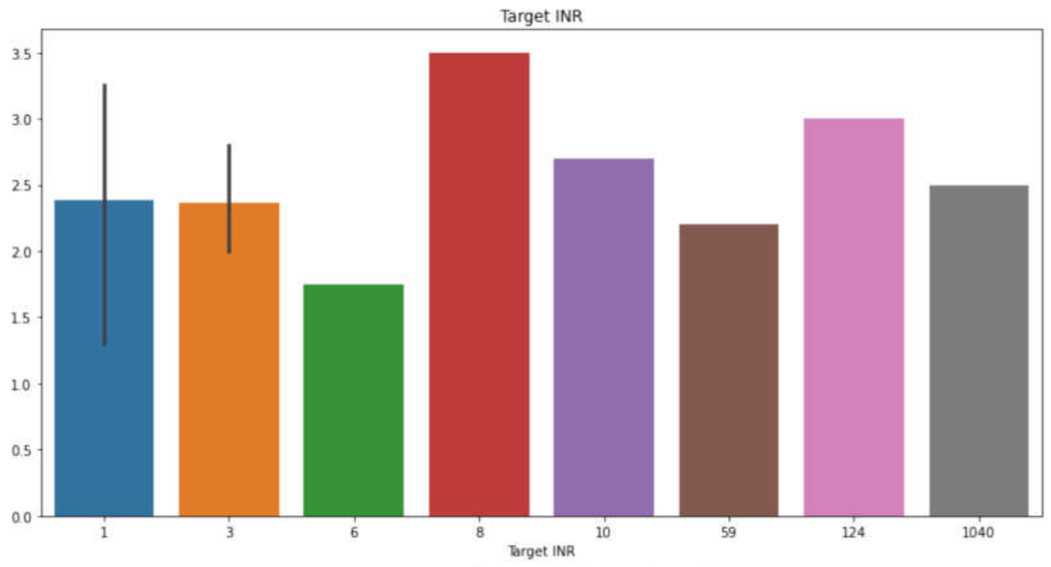

Weight (kg)
100
80
60
40
20
0
Weight (kg)

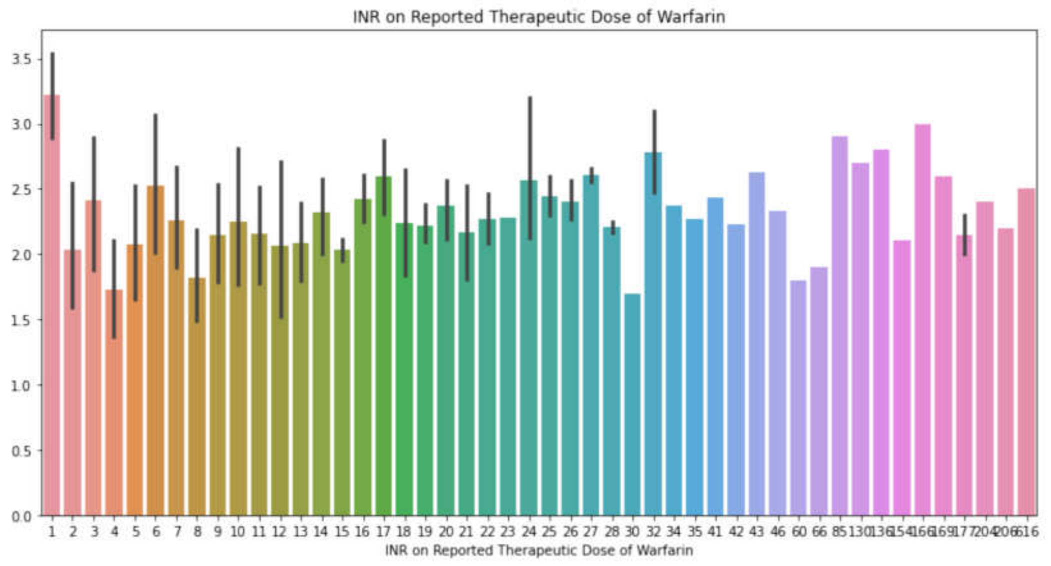

Diabetes
1.0
0.8
0.6
0.4
0.2
0.0
614
2669
Diabetes

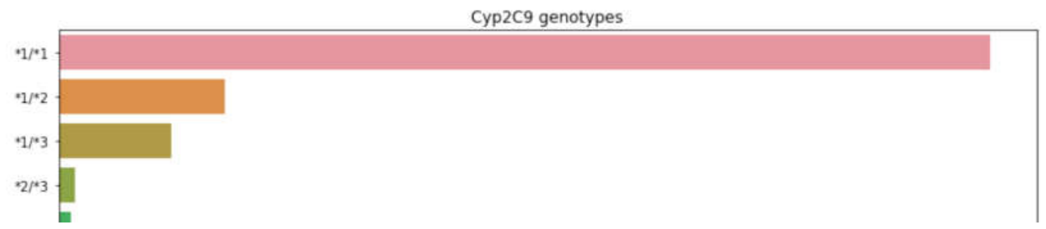

Simvastatin (Zocor)
1.0
0.8
0.6
0.4
0.2
0.0

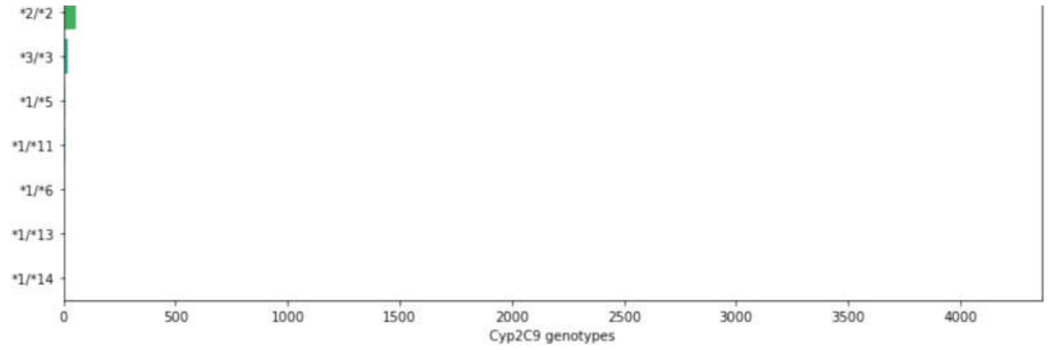
565
3296
Simvastatin (Zocor)

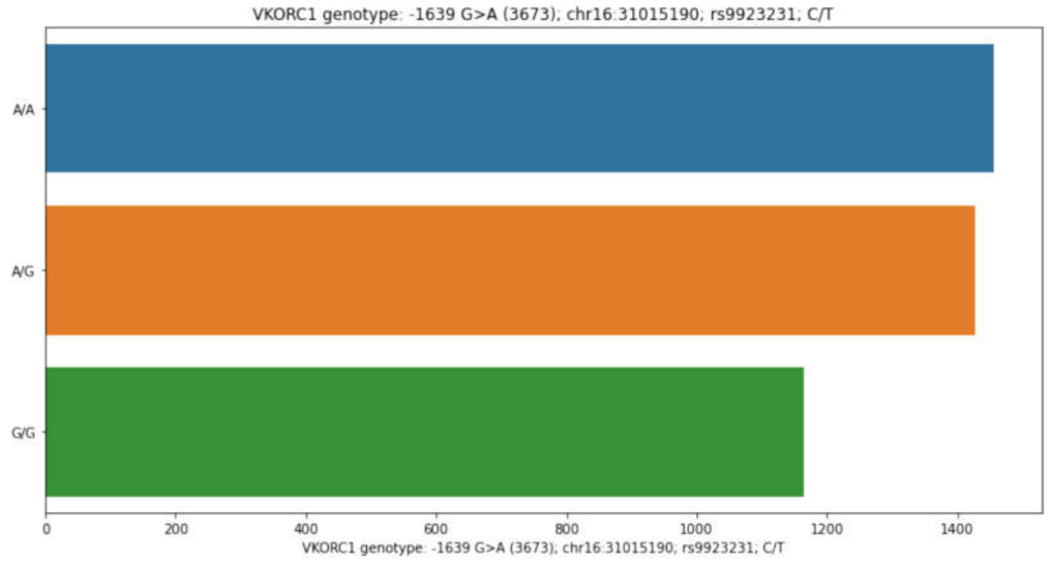

Amiodarone (Cordarone)
1.0
0.8
0.6
0.4
0.2
0.0
277
3905
Amiodarone (Cordarone)

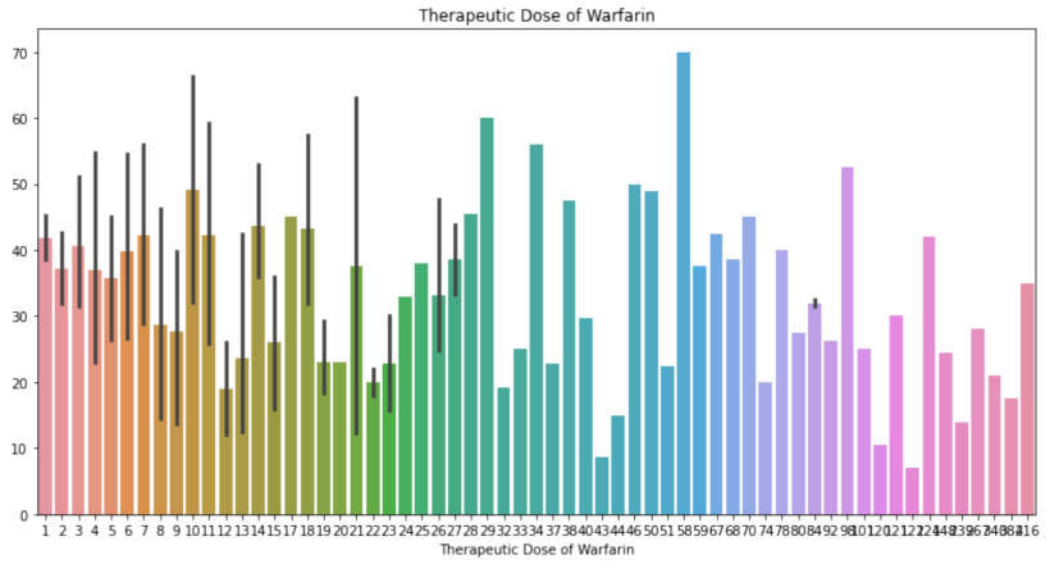

Target INR
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1
3
6
8
10
59
124
1040
Target INR


INR on Reported Therapeutic Dose of Warfarin
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 30 32 34 35 41 42 43 46 60 66 85 130 136 154 166 169 177 204 206 616
INR on Reported Therapeutic Dose of Warfarin


Cyp2C9 genotypes
*1/*1
*1/*2
*1/*3
*2/*3



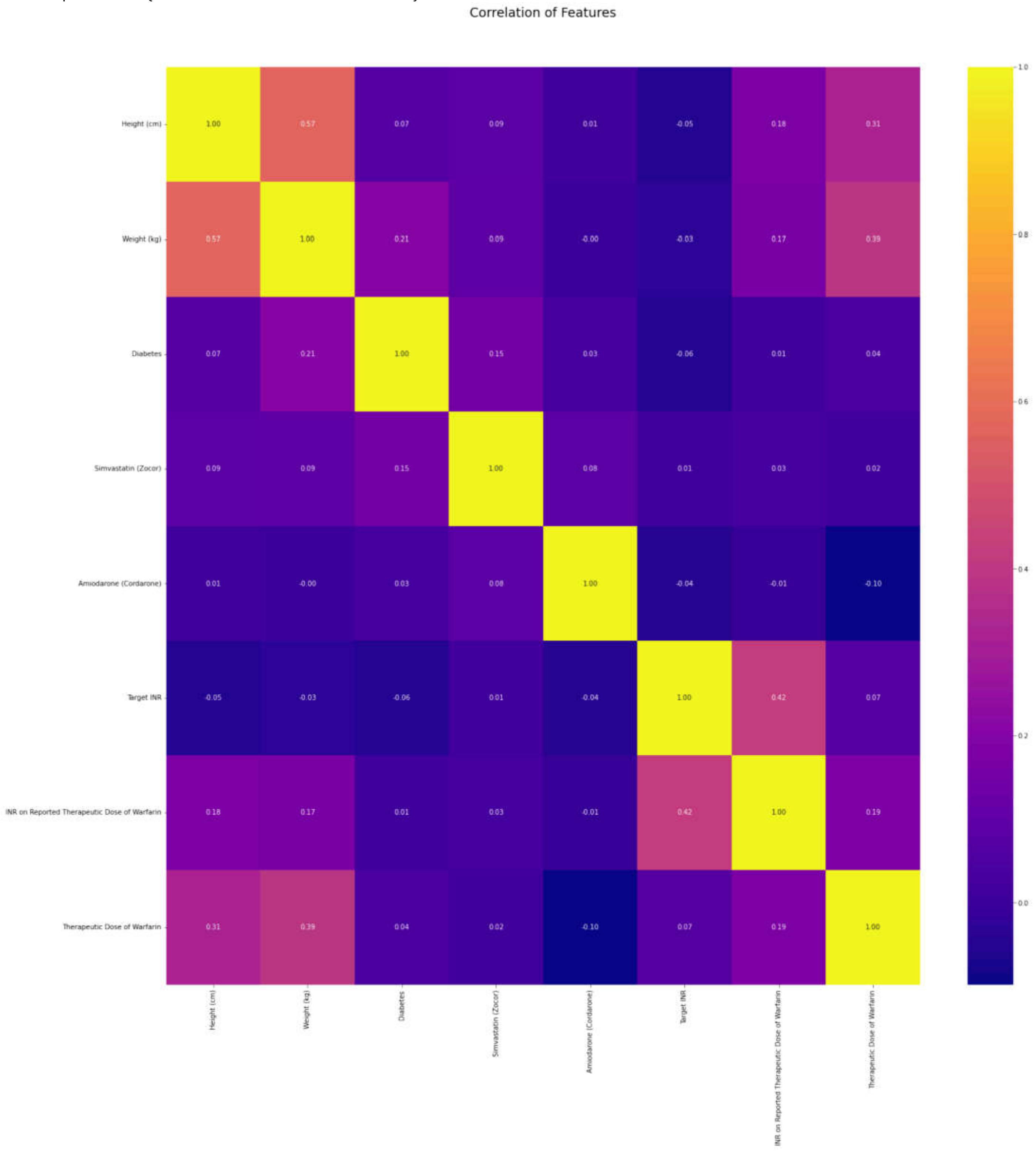




dataset Correlation represented by confusion matrix

```
# Confusion matrix of the data to see correlation
plt.figure(figsize = (25,25))
plt.title('Correlation of Features', y=1.05, size=19)
sns.heatmap(data.corr(), cmap='plasma',annot=True, fmt='.2f')
```

```
<AxesSubplot:title={'center':'Correlation of Features'}>
```



	Height (cm)	Weight (kg)	Diabetes	Simvastatin (Zocor)	Amiodarone (Cordarone)	Target INR	INR on Reported Therapeutic Dose of Warfarin	Therapeutic Dose of Warfarin
Height (cm)	1.00	0.57	0.07	0.09	0.01	-0.05	0.18	0.31
Weight (kg)	0.57	1.00	0.21	0.09	-0.00	-0.03	0.17	0.39
Diabetes	0.07	0.21	1.00	0.15	0.03	-0.06	0.01	0.04
Simvastatin (Zocor)	0.09	0.09	0.15	1.00	0.08	0.01	0.03	0.02
Amiodarone (Cordarone)	0.01	-0.00	0.03	0.08	1.00	-0.04	-0.01	-0.10
Target INR	-0.05	-0.03	-0.06	0.01	-0.04	1.00	0.42	0.07
INR on Reported Therapeutic Dose of Warfarin	0.18	0.17	0.01	0.03	-0.01	0.42	1.00	0.19
Therapeutic Dose of Warfarin	0.31	0.39	0.04	0.02	-0.10	0.07	0.19	1.00

Data Classification Using KNN and Decision Tree

In python Machine learning is done easier using libraries like sklearn which provides the necessary tools to analyse the data and create prediction models. In this task KNN and Decision Tree models were selected for classification and hence will call these models,will also call metrics class function which will help obtain report accuracy, precision, recall, F1-score and ROC curves (AUC-ROC) analysis.

```
# import pipeline
from sklearn.pipeline import Pipeline
#import sklearn libraries and models
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score, GridSearchCV
from sklearn.metrics import  classification_report, accuracy_score, roc_curve
```

Convert categorical datatype from object to numerical using labelencoder

```
# Using label encoder that converts all Dtypes into an uniform format
lb_make = LabelEncoder()

Objt = [col for col in data.columns if data[col].dtype=="O"]
for item in Objt:
    data[item] = lb_make.fit_transform(data[item])
data.reset_index()
data.reset_index(drop=True)
data
```

	Gender	Race (Reported)	Age	Height (cm)	Weight (kg)	Diabetes	Simvastatin (Zocor)	Amiodarone (Cordarone)	Target INR	INR on Reported Therapeutic Dose of Warfarin	Cyp2C9 genotypes	VKORC1 genotype: -1639 G>A (3673); chr16:31015190; rs9923231; C/T	Thera
0	1	20	5	193.040	115.70	NaN	0.0	0.0	2.5	2.60	0	1	
1	0	20	4	176.530	144.20	NaN	0.0	0.0	2.5	2.15	0	0	
2	0	20	3	162.560	77.10	NaN	0.0	0.0	2.5	1.90	0	2	
3	1	20	5	182.245	90.70	NaN	0.0	0.0	2.5	2.40	0	1	
4	1	20	4	167.640	72.60	NaN	0.0	0.0	2.5	1.90	5	1	
...	...	...	...	...	...	...	...	...	...	...	...	...	
5695	1	20	1	185.420	113.64	0.0	0.0	0.0	NaN	2.80	0	3	
5696	0	20	6	160.020	55.91	0.0	0.0	0.0	NaN	2.80	5	3	
5697	1	20	5	187.960	97.73	0.0	0.0	0.0	NaN	2.00	0	3	
5698	1	20	5	177.800	87.27	0.0	0.0	0.0	NaN	2.00	11	3	
5699	1	20	6	190.500	79.55	0.0	0.0	0.0	NaN	3.00	11	3	

```
# replace all the nah with a zero
data = data.fillna(value=0)
```

```
#The data is now ready after filling all NaN values with 0
# first class is those with >30 mg/wk (high required dose (HRD)) and
# the second class contains the patients who need doses of ≤30 mg/wk (low required dose (LRD)).
# we create these classes in the Therapeutic Dose of Warfarin column
y = data['Therapeutic Dose of Warfarin']
for i in range(len(y)):
    if y[i] <= 30.0:
        y[i] = 0
    if y[i] > 30:
        y[i] = 1
```

Split the data

```
# define features and label
X = data.drop(["Therapeutic Dose of Warfarin"], axis='columns')
y = data['Therapeutic Dose of Warfarin']
```

```
# split training and testing data with 30% for testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,random_state=0)
```

KNN Classifier with GridSearchCv tuning

```
knn = KNeighborsClassifier()
k_range = list(range(1, 30))
paramz = dict(n_neighbors=k_range)
# defining parameter range
gsvk = GridSearchCV(knn, paramz, cv=10, scoring='accuracy')
# fitting the model for grid search
gsvk.fit(X_train, y_train)

best_parameters = gsvk.best_params_
print(best_parameters)
```

```
{'n_neighbors': 29}
```

```
# best parameters were found to be 29 neighbors
knn = KNeighborsClassifier(n_neighbors = 29)
knn.fit(X_train, y_train)
y_predk=knn.predict(X_test)
print("Accuracy Score:",accuracy_score(y_test, y_predk))
```

```
Accuracy Score: 0.704093567251462
```

```
print("Classification Report for KNN Classifier")
print(classification_report(y_test, y_predk))
```

```
Classification Report for KNN Classifier
              precision    recall  f1-score   support

         0.0       0.74      0.75      0.75       991
         1.0       0.65      0.63      0.64       719

    accuracy                           0.70      1710
   macro avg       0.70      0.69      0.70      1710
weighted avg       0.70      0.70      0.70      1710
```

```
print("ROC curves for KNN Classifier")
print(roc_curve(y_test, y_predk))
```

```
ROC curves for KNN Classifier
(array([0.        , 0.24520686, 1.        ]), array([0.        , 0.63421419, 1.        ]), array([2., 1., 0.]))
```

Decision Tree with GridSearchCv tuning

```
params = {
    'criterion': ['gini', 'entropy']
}
```

```
# Create the gridSearch Model
gsvr = GridSearchCV(estimator=DecisionTreeClassifier(),
                     param_grid=params,
                     scoring='accuracy',
                     cv=10)
```

```
# train the model
gsvr.fit(X_train,y_train)
```

```
▸ GridSearchCV
▸ estimator: DecisionTreeClassifier
    ▸ DecisionTreeClassifier
```

```
best_parameters = gsvr.best_params_
print(best_parameters)
```

```
{'criterion': 'entropy'}
```

```
# best parameters were found to be entropy
tree = DecisionTreeClassifier(criterion= 'entropy')
tree.fit(X_train, y_train)
y_predT=tree.predict(X_test)
print("Accuracy Score:",accuracy_score(y_test, y_predT))
```

```
Accuracy Score: 0.6707602339181287
```

```
print("Classification Report for DecisionTreeClassifier")
print(classification_report(y_test, y_predT))
```

```
Classification Report for DecisionTreeClassifier
              precision    recall  f1-score   support

         0.0       0.72      0.71      0.71       991
         1.0       0.61      0.62      0.61       719

    accuracy                           0.67      1710
   macro avg       0.66      0.66      0.66      1710
weighted avg       0.67      0.67      0.67      1710
```

```
print("ROC curves for DecisionTreeClassifier")
print(roc_curve(y_test, y_predT))
```

```
ROC curves for DecisionTreeClassifier
(array([0.        , 0.29061554, 1.        ]), array([0.        , 0.61752434, 1.        ]), array([2., 1., 0.]))
```

Analysis Done Using Artificial Neural Network

At this point will build a neural network to help us analyse the data and train a model to predict dosage. tensorflow is a library used in creation of neural networks in python and in this task we will import the necessary modules. using sequential(), will create an input layer with 12 nodes like our columns after will have two hidden layers of 12 and 8 nodes respectively and will use relu activation function. The last layer is the output layer of a single node and use sigmoid activation function.

```
# import the necessary libraries
import numpy as np
import tensorflow as tf
```

```
# need to standardize the dataset
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

Build the ANN