Project title:Prostate Cancer Diagnosis using Clinical and Biopsy Images

Details of the Team

Team no.	MLB-07	
Div:	B	
Sl. No.	Name	SRN.
1	Chandsab Engineer	02FE21BCS022
2	Radhika Khot	02FE21BCS067
3	Rashmi Shinde	02FE21BCS113

Dataset discription

Directory: Prostate Main dataset contains nearly all the PLCO study data available for prostate cancer screening, incidence, and mortality analyses. The dataset contains one record for each of the approximately 77,000 male participants in the PLCO trial And it contain 214 columns. Again we have 6 more dataset whcih include diagnosis ,screening ,medical complication ,treatment and pathology images information.

EDA

In [5]:
```
1 df_main=pd.read_csv(r"F:\5th sem\project\Prostate\pros_data_mar22_d032222.
```

In [7]:
```
1 df_main
```

Out[7]:

	reasfollp	reassympp	reassurvp	reasothp	pros_cancer	pros_dx_psa	pros_dx_psa_gap	i
0	NaN	NaN	NaN	NaN	0	NaN	NaN	
1	NaN	NaN	NaN	NaN	0	NaN	NaN	
2	NaN	NaN	NaN	NaN	0	NaN	NaN	
3	NaN	NaN	NaN	NaN	0	NaN	NaN	
4	NaN	NaN	NaN	NaN	0	NaN	NaN	
...	...	...	...	...	...	...	...	
76673	NaN	NaN	NaN	NaN	0	NaN	NaN	
76674	NaN	NaN	NaN	NaN	0	NaN	NaN	
76675	NaN	NaN	NaN	NaN	0	NaN	NaN	
76676	0.0	0.0	NaN	1.0	1	2.2	12.0	
76677	NaN	NaN	NaN	NaN	0	NaN	NaN	

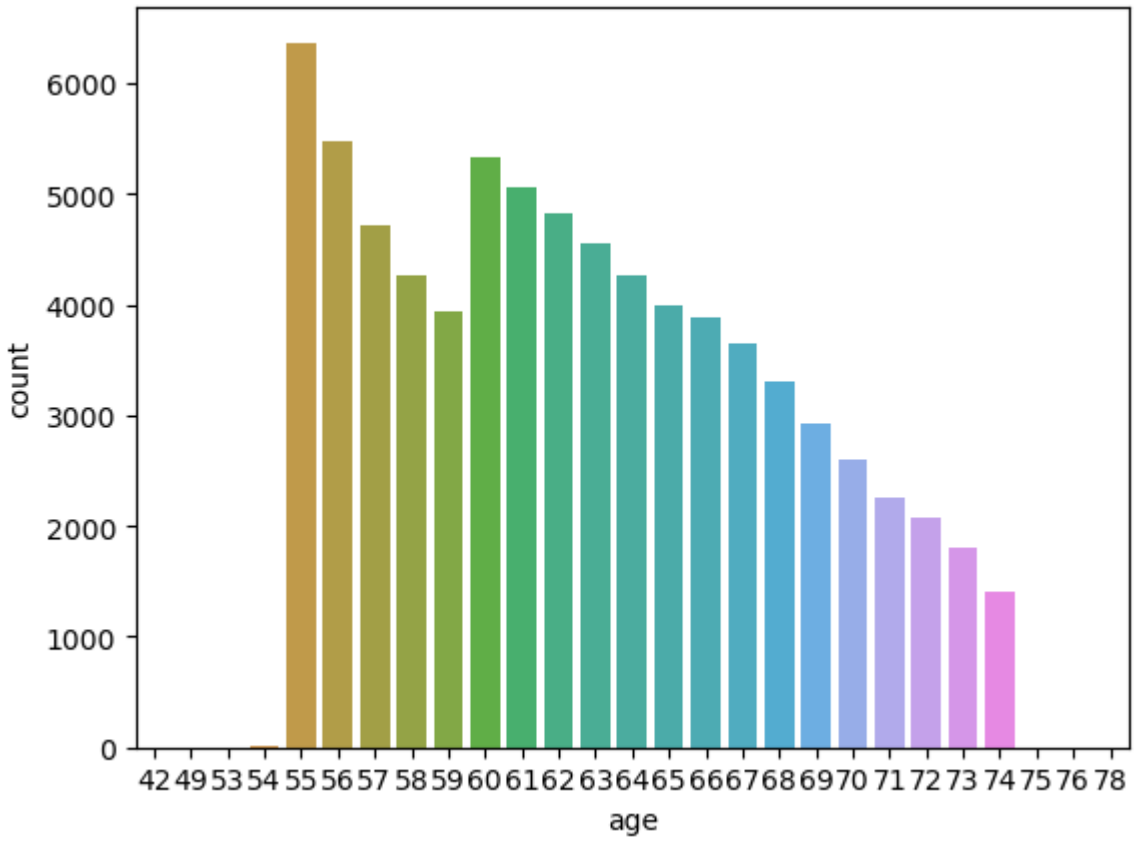
76678 rows × 214 columns

In [11]:
```
1 df_main["age"].value_counts()
```

Out[11]:
```
55    6368
56    5473
60    5331
61    5054
62    4823
57    4722
63    4548
64    4266
58    4256
65    3996
59    3931
66    3881
67    3650
68    3305
69    2928
70    2596
71    2256
72    2079
73    1800
74    1400
54       6
78       2
53       2
75       2
42       1
49       1
76       1
Name: age, dtype: int64
```

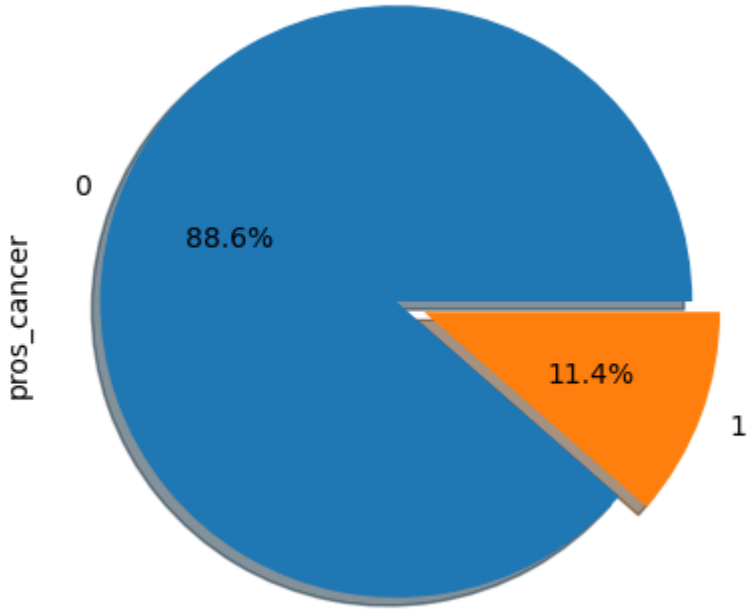
1.Which age group people were more in count

In [13]:
```
sns.countplot(x='age',data=df_main)
plt.show()
```



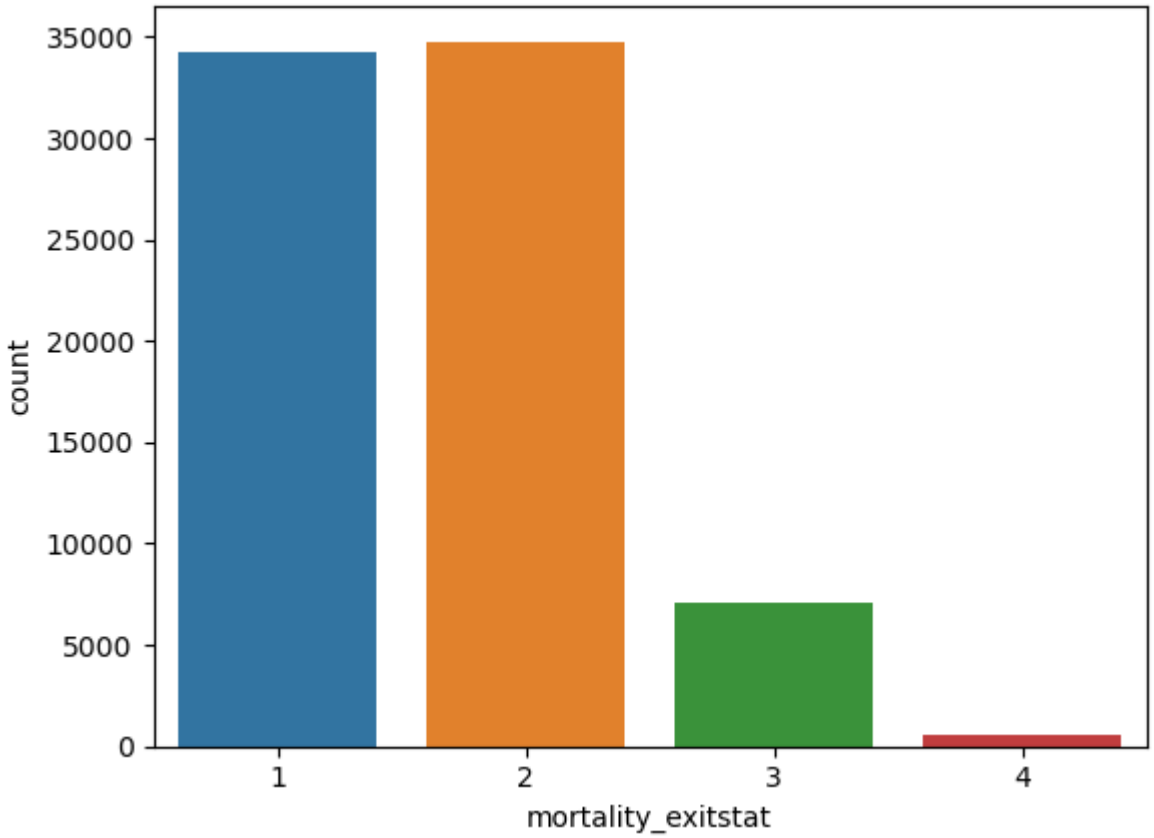
2.How many participants are confirmed with prostate cancer.

In [14]:
```
df_main["pros_cancer"].value_counts().plot.pie(explode=[0.1, 0], autopct="
plt.show()
```



3. how many are participant are dead because of prostate cancer and what was their prostate stage

In [17]:
```
sns.countplot(x='mortality_exitstat',data=df_main)
plt.show()
```



4.How did family history and bad habits affect the participant for cause of prostate cancer

In [26]:
```
table = df_main.groupby(["pros_cancer"])[["cig_stat"]].count()
table
```

Out[26]:

pros_cancer	cig_stat
0	65294
1	8536

In [33]:
```
table = df_main.groupby(["pros_cancer"])[["cig_stat"]].count()
table
```

Out[33]:

pros_cancer	cig_stat
0	65294
1	8536

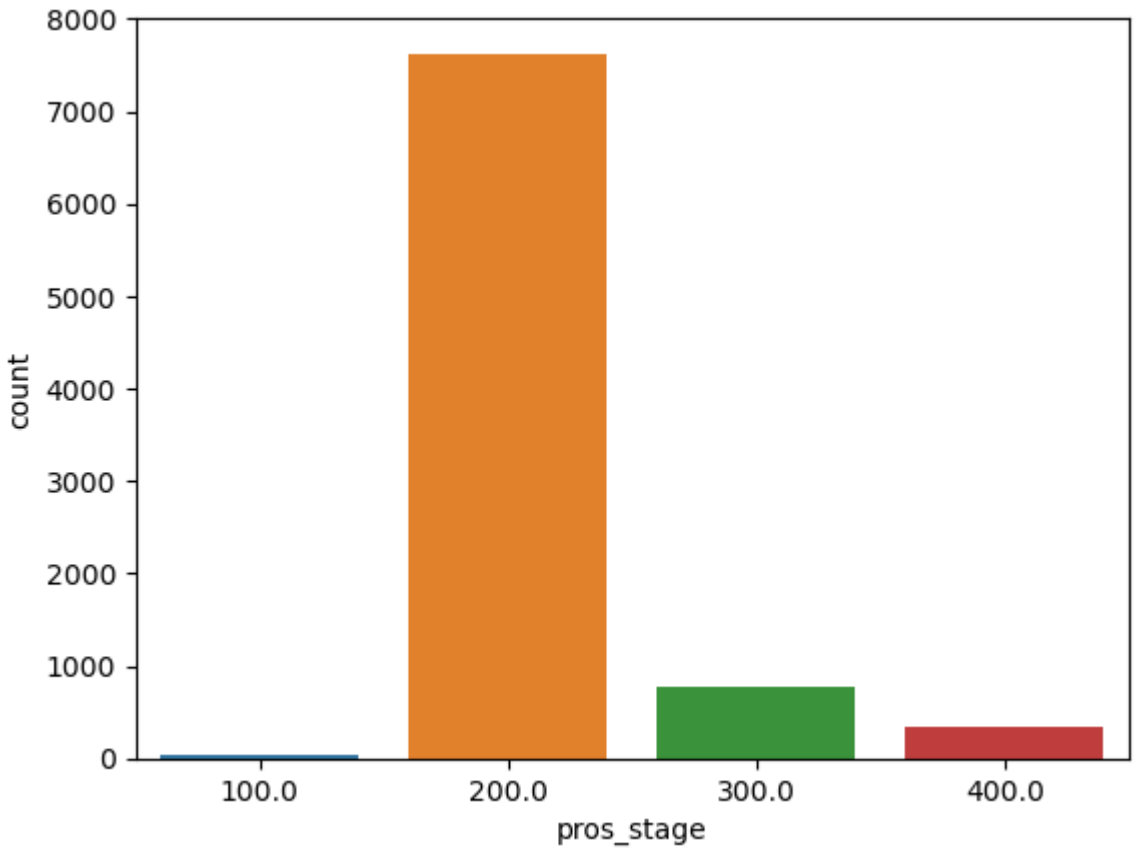
In [35]:
```
table = df_main.groupby(["pros_stage"])[["pros_cancer"]].count()
table
```

Out[35]:

pros_stage	pros_cancer
100.0	37
200.0	7624
300.0	766
400.0	341

5.Which stage of prostate cancer affected the majority of individuals.

In [32]:
```
sns.countplot(x='pros_stage',data=df_main)
plt.show()
```



preprocessing

In [6]:

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassi
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import OrdinalEncoder, MinMaxScaler
from sklearn.metrics import mean_absolute_error, confusion_matrix, classif
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
```

1.Data Loading:

In [69]:

```
screen_df=pd.read_csv('F:\pros_screen_data_mar22_d032222.csv')
screen_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 177314 entries, 0 to 177313
Data columns (total 80 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   study_yr                177314 non-null  int64
 1   examinerid_pvis1        128075 non-null  float64
 2   examinerid_pvis2        201 non-null     float64
 3   examinerid_pvis3        0 non-null       float64
 4   dreres_pvis1            128075 non-null  float64
 5   dreres_pvis2            201 non-null     float64
 6   dreres_pvis3            0 non-null       float64
 7   inad_dis_p1             301 non-null     float64
 8   inad_dis_p2             1 non-null       float64
 9   inad_dis_p3             0 non-null       float64
 10  inad_ref_p1             15 non-null      float64
 11  inad_ref_p2             0 non-null       float64
 12  inad_ref_p3             0 non-null       float64
 13  inad_oth_p1             2447 non-null    float64
```

In [70]:
```
screen_df['study_yr'].value_counts()
```

Out[70]:
```
0    34258
1    32694
2    31697
3    30546
5    25949
4    22170
Name: study_yr, dtype: int64
```

In [71]:
```
screen_df_3 = screen_df.query('study_yr==3')
```

In [72]:
```
screen_ab_df=pd.read_csv("F:\pros_scrsub_data_mar22_d032222.csv")
```

In [73]:
```
screen_ab_df[screen_ab_df.duplicated(['plco_id'])]
```

Out[73]:

	VISIT	study_yr	source	sbcd	loc_1	loc_2	loc_3	loc_4	loc_5	loc_6	loc_7	loc_8	
1	1	3	DRE	1	NaN	NaN	NaN	1.0	NaN	1.0	NaN	NaN	
6	1	1	DRE	1	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	
7	1	2	DRE	1	NaN	NaN	NaN	NaN	1.0	1.0	NaN	NaN	
8	1	3	DRE	1	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	
10	1	3	DRE	1	NaN	1.0	NaN	1.0	NaN	1.0	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	

In [74]:
```
screen_ab_df_3= screen_ab_df.query('study_yr==3')
```

In [75]:
```
screen_ab_df_31=screen_ab_df_3.query('sbcd==1')
```

In [76]:
```
screen_ab_df_31E[screen_ab_df_31E.duplicated(['plco_id'])]
```

Out[76]:

VISIT	study_yr	source	sbcd	loc_1	loc_2	loc_3	loc_4	loc_5	loc_6	loc_7	loc_8	extent	s

In [77]:
```
screen_ab_df_31E=screen_ab_df_31.query('source=="DRE"')
```

In [78]:
```
screen_ab_df_3= screen_ab_df.query('study_yr==3')
```

In [79]:
```
df_screen=pd.merge(screen_df_3, screen_ab_df_31E, how='outer')
```

In [80]:
```
df_med=pd.read_csv("F:\pros_screen_data_mar22_d032222.csv")
```

In [81]:
```
df_med
```

Out[81]:

	study_yr	examinerid_pvis1	examinerid_pvis2	examinerid_pvis3	dreres_pvis1	dreres_p
0	0	44405.0	NaN	NaN	3.0	
1	1	44409.0	NaN	NaN	4.0	
2	2	44405.0	NaN	NaN	3.0	
3	3	44404.0	NaN	NaN	1.0	
4	4	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	
177309	1	90201.0	NaN	NaN	3.0	
177310	2	90219.0	NaN	NaN	3.0	
177311	3	90201.0	NaN	NaN	3.0	
177312	4	NaN	NaN	NaN	NaN	
177313	5	NaN	NaN	NaN	NaN	

In [82]:
```
df_medical=df_med.query('study_yr==3')
df_medical[df_medical.duplicated('plco_id')]
df_s_m1=pd.merge(df_screen,df_medical,how='outer',on='plco_id')
df_s_m=pd.merge(df_screen,df_medical,how='outer')
```

In [83]:
```
df_trt=pd.read_csv("F:\pros_trt_data_mar22_d032222.csv")
df_trt[df_trt.duplicated('plco_id')]
df_trt=df_trt.drop_duplicates(['plco_id'] ,keep='first' )
```

2.Data Integeration:Merging of dataset

We have merged dataset taking common study year as 3rd and We divided the data into two subsets. One subset is used for diagnosis and another is used for staging.

In [84]:
```
df1 = pd.merge(df,df_trt,how='outer')
```

In [85]:
```
df_main=pd.read_csv(r"F:\5th sem\project\Prostate\pros_data_mar22_d032222.
df=pd.merge(df_main, df_s_m, how='outer')
```

In [86]:
```
1 df1
```

0	NaN	NaN	NaN	NaN	0	NaN	NaN
1	NaN	NaN	NaN	NaN	0	NaN	NaN
2	NaN	NaN	NaN	NaN	0	NaN	NaN
3	NaN	NaN	NaN	NaN	0	NaN	NaN
4	NaN	NaN	NaN	NaN	0	NaN	NaN
...	...	...	...	...	...	...	...
76673	NaN	NaN	NaN	NaN	0	NaN	NaN
76674	NaN	NaN	NaN	NaN	0	NaN	NaN
76675	NaN	NaN	NaN	NaN	0	NaN	NaN
76676	0.0	0.0	NaN	1.0	1	2.2	12.0
76677	NaN	NaN	NaN	NaN	0	NaN	NaN

76678 rows × 309 columns

In [87]:

```python
from tabulate import tabulate

# Summary statistics
summary_stats = df_main.describe()

# Number of non-null values in each column
non_null_count = df_main.count()

# Data types of each column
data_types = df_main.dtypes

# Unique values in each column
unique_values = df_main.nunique()

# Missing values in each column
mv=df_main.isnull().sum()

# Mode (most frequent value) of each column
modes = df_main.mode().iloc[0]


# Combine all the information into a dataset description DataFrame
data_description = pd.DataFrame({
    'Data Types': data_types,
    'Non-Null Count': non_null_count,
    'Unique Values': unique_values,
    'Missing Values':mv,
    'Mode': modes,
})

# Print the dataset description in a nice table format
print("Dataset Description:")
print(tabulate(data_description, headers='keys', tablefmt='pretty'))
```

Dataset Description:

	Data Types	Non-Null Count	Unique Values	Missing Values	Mode
reasfollp	float64	8777	2	67901	0.0
reassympp	float64	8777	2	67901	0.0
reassurvp	float64	1576	2	75102	0.0
reasothp	float64	8777	2	67901	1.0
pros_cancer	int64	76678	2	0	0.0
pros_dx_psa	float64	8259	1348	68419	5.2
pros_dx_psa_gap	float64	8259	548	68419	0.0
intstatp_cat	int64	76678	6	0	0.0
pros_stage	float64	8768	4	67910	200.0
pros_stage_7e	float64	8768	6	67910	210.0
pros_stage_t	float64	8762	17	67916	130.0
pros_stage_n	float64	8748	4	67930	0.0
pros_stage_m	float64	8753	5	67925	0.0
pros_clinstage	float64	8680	4	67998	200.0
pros_clinstage_7e	float64	8683	5	67995	100.0
pros_clinstage_t	float64	8719	14	67959	130.0
pros_clinstage_n	float64	8696	4	67982	0.0
pros_clinstage_m	float64	8707	5	67971	0.0
pros_pathstage	float64	3172	3	73506	200.0
pros_pathstage_7e	float64	3172	5	73506	210.0
pros_pathstage_t	float64	3214	16	73464	220.0
pros_pathstage_n	float64	3209	4	73469	0.0
pros_pathstage_m	float64	3215	4	73463	0.0
pros_gleason	float64	8654	9	68024	6.0
pros_gleason_source	float64	8776	3	67902	2.0

pros_gleason_biop	float64	8608	9	68070	6.0
pros_gleason_prost	float64	3195	9	73483	7.0
pros_topography	object	8776	1	67902	C619
pros_grade	float64	8776	5	67902	2.0
pros_behavior	float64	8776	1	67902	3.0
pros_morphology	float64	8776	13	67902	8140.0
curative_prostp	float64	8729	2	67949	0.0
curative_hormp	float64	8729	2	67949	0.0
curative_radp	float64	8729	2	67949	0.0
curative_othp	float64	8729	2	67949	0.0
neoadjuvantp	float64	8776	2	67902	0.0
primary_trtp	int64	76678	8	0	0.0
pros_exitstat	int64	76678	9	0	8.0
pros_exitage	int64	76678	36	0	69.0
pros_cancer_first	float64	8776	2	67902	1.0
pros_num_heslide_imgs	float64	1095	6	75583	3.0
pros_has_deliv_heslide_img	float64	1095	1	75583	1.0
pros_seer	float64	8776	1	67902	28010.0
pros_annyr	float64	8776	16	67902	0.0
plco_id	object	76678	76678	0	A-000899-7
build	object	76678	1	0	mar22/03.22.22
build_cancers	int64	76678	1	0	1.0
build_incidence_cutoff	int64	76678	1	0	1.0
primary_trtp_days	float64	7662	3816	69016	132.0
pros_exitdays	int64	76678	5870	0	0.0
pros_cancer_diagdays	float64	8776	4038	67902	70.0
biopplink0	int64	76678	2	0	0.0
biopplink1	int64	76678	2	0	0.0
biopplink2	int64	76678	2

0	biopplink3	int64	76678	2	0
0.0	biopplink4	int64	76678	2	0
0.0	biopplink5	int64	76678	2	0
0.0	pros_mra_stat0	int64	76678	4	0
0.0	pros_mra_stat1	int64	76678	4	0
0.0	pros_mra_stat2	int64	76678	4	0
0.0	pros_mra_stat3	int64	76678	4	0
0.0	pros_mra_stat4	int64	76678	4	0
0.0	pros_mra_stat5	int64	76678	4	0
0.0	psa_result0	float64	38340	5	38338
1.0	psa_result1	float64	38340	5	38338
1.0	psa_result2	float64	38340	5	38338
1.0	psa_result3	float64	38340	5	38338
1.0	psa_result4	float64	38340	5	38338
1.0	psa_result5	float64	38340	5	38338
1.0	psa_level0	float64	34224	1247	42454
0.53	psa_level1	float64	32660	1115	44018
0.58	psa_level2	float64	31655	1124	45023
0.56	psa_level3	float64	30489	1124	46189
0.54	psa_level4	float64	22158	983	54520
0.56	psa_level5	float64	25935	1078	50743
0.57	dre_result0	float64	38340	6	38338
1.0	dre_result1	float64	38340	6	38338
3.0	dre_result2	float64	38340	6	38338
3.0	dre_result3	float64	38340	6	38338
3.0	psa_prot	float64	38340	3	38338
3.0	psa_days0	float64	34246	284	42432
21.0	psa_days1	float64	32687	373	43991
371.0					

psa_days2	float64	31690	388	44988	721.0
psa_days3	float64	30532	410	46146	1092.0
psa_days4	float64	22170	405	54508	1448.0
psa_days5	float64	25949	547	50729	1813.0
dre_days0	float64	34128	286	42550	21.0
dre_days1	float64	32447	371	44231	371.0
dre_days2	float64	31450	388	45228	721.0
dre_days3	float64	30243	410	46435	1092.0
educat	float64	73642	7	3036	7.0
marital	float64	73642	5	3036	1.0
occupat	float64	73492	7	3186	4.0
pipe	float64	73372	3	3306	0.0
cigar	float64	73394	3	3284	0.0
sisters	float64	73222	8	3456	1.0
brothers	float64	73454	8	3224	1.0
asp	float64	73245	2	3433	1.0
ibup	float64	73520	2	3158	0.0
bq_adminm	float64	73314	5	3364	1.0
asppd	float64	73619	8	3059	0.0
ibuppd	float64	73467	8	3211	0.0
rectal_history	float64	73831	4	2847	0.0
urinatea	float64	25661	6	51017	4.0
enlprosa	float64	15997	6	60681	4.0
infprosa	float64	5177	6	71501	4.0
vasecta	float64	19971	4	56707	3.0
hyperten_f	float64	73421	2	3257	0.0
hearta_f	float64	73385	2	3293	0.0
stroke_f	float64	73387	2	3291	0.0
emphys_f	float64	73377	2

3301	0.0				
bronchit_f	float64	73349	2		
3329	0.0				
diabetes_f	float64	73388	2		
3290	0.0				
polyps_f	float64	73300	2		
3378	0.0				
arthrit_f	float64	73358	2		
3320	0.0				
osteopor_f	float64	73299	2		
3379	0.0				
divertic_f	float64	73282	2		
3396	0.0				
gallblad_f	float64	73318	2		
3360	0.0				
bq_returned	int64	76678	2		
0	1.0				
bq_age	float64	73854	28		
2824	55.0				
race7	int64	76678	7		
0	1.0				
hispanic_f	float64	71758	2		
4920	0.0				
surg_biopsy	float64	71713	2		
4965	0.0				
surg_resection	float64	71522	2		
5156	0.0				
surg_prostatectomy	float64	71480	2		
5198	0.0				
surg_age	float64	5707	5		
70971	4.0				
surg_any	float64	73806	3		
2872	0.0				
enlpros_f	float64	73717	2		
2961	0.0				
infpros_f	float64	61621	2		
15057	0.0				
prosprob_f	float64	73688	2		
2990	0.0				
urinate_f	float64	73715	6		
2963	1.0				
vasect_f	float64	73593	2		
3085	0.0				
smoked_f	float64	73831	2		
2847	1.0				
smokea_f	float64	46561	62		
30117	18.0				
rsmoker_f	float64	46913	2		
29765	0.0				
ssmokea_f	float64	37577	67		
39101	40.0				
cigpd_f	float64	73722	8		
2956	0.0				
filtered_f	float64	46813	3		
29865	1.0				
cig_stat	float64	73830	3		
2848	2.0				

cig_stop	float64	46207	63	30471	0.0
cig_years	float64	72813	66	3865	0.0
pack_years	float64	72731	219	3947	0.0
bmi_20	float64	72417	1711	4261	21.5204081632653
bmi_50	float64	72739	2121	3939	25.1071428571429
bmi_curr	float64	72636	2620	4042	25.8244897959184
bmi_curc	float64	72636	4	4042	3.0
weight_f	float64	73131	260	3547	180.0
weight20_f	float64	72902	193	3776	150.0
weight50_f	float64	73235	238	3443	180.0
height_f	float64	73123	33	3555	70.0
psa_history	float64	73834	4	2844	0.0
bmi_20c	float64	72417	4	4261	2.0
bmi_50c	float64	72739	4	3939	3.0
colon_comorbidity	float64	73102	2	3576	0.0
liver_comorbidity	float64	73286	2	3392	0.0
fh_cancer	float64	73616	2	3062	1.0
pros_fh	float64	73197	3	3481	0.0
pros_fh_cnt	float64	73197	5	3481	0.0
pros_fh_age	float64	5193	73	71485	70.0
bq_compdays	float64	73854	675	2824	0.0
d_dthp	int64	76678	2	0	0.0
f_dthp	int64	76678	2	0	0.0
d_codeath_cat	float64	34241	14	42437	100.0
f_codeath_cat	float64	34240	14	42438	100.0
d_cancersite	float64	34241	17	42437	999.0
f_cancersite	float64	34240	17	42438	999.0
d_seer_death	float64	34241	68	42437	50060.0
f_seer_death	float64	34240	68		

```
42438      |     50060.0      |
|      is_dead_with_cod      |   int64    |     76678      |       2        |
0        |       0.0        |
|          is_dead           |   int64    |     76678      |       2        |
0        |       0.0        |
|     mortality_exitage      |   int64    |     76678      |       45       |
0        |       78.0       |
|     mortality_exitstat     |   int64    |     76678      |       4        |
0        |       2.0        |
|     build_death_cutoff     |   int64    |     76678      |       1        |
0        |       4.0        |
|          dth_days          |  float64   |     34254      |      8441      |
42424      |      6762.0      |
|     mortality_exitdays     |   int64    |     76678      |      8847      |
0        |      8138.0      |
|        entryage_bq         |  float64   |     73854      |       28       |
2824       |       55.0       |
|        entryage_dqx        |  float64   |     31720      |       22       |
44958      |       56.0       |
|        entryage_dhq        |  float64   |     56337      |       30       |
20341      |       59.0       |
|        entryage_sqx        |  float64   |     49323      |       29       |
27355      |       66.0       |
|        entryage_muq        |  float64   |     28605      |       29       |
48073      |       72.0       |
|         ph_any_bq          |  float64   |     73854      |       3        |
2824       |       0.0        |
|         ph_any_dqx         |  float64   |     31754      |       3        |
44924      |       0.0        |
|         ph_any_dhq         |  float64   |     57484      |       3        |
19194      |       0.0        |
|         ph_any_sqx         |  float64   |     49323      |       3        |
27355      |       0.0        |
|         ph_any_muq         |  float64   |     28605      |       3        |
48073      |       0.0        |
|         ph_pros_bq         |  float64   |     73854      |       2        |
2824       |       0.0        |
|        ph_pros_dqx         |  float64   |     31754      |       2        |
44924      |       0.0        |
|        ph_pros_dhq         |  float64   |     57484      |       2        |
19194      |       0.0        |
|        ph_pros_sqx         |  float64   |     49323      |       2        |
27355      |       0.0        |
|        ph_pros_muq         |  float64   |     28605      |       2        |
48073      |       0.0        |
|        ph_any_trial        |   int64    |     76678      |       3        |
0        |       0.0        |
|       ph_pros_trial        |   int64    |     76678      |       2        |
0        |       0.0        |
|      pros_eligible_bq      |   int64    |     76678      |       2        |
0        |       1.0        |
|     pros_eligible_sqx      |   int64    |     76678      |       2        |
0        |       1.0        |
|     pros_eligible_dhq      |   int64    |     76678      |       2        |
0        |       1.0        |
|     pros_eligible_dqx      |   int64    |     76678      |       2        |
0        |       0.0        |
```

```
|        entrydays_bq        |  float64   |     73854      |      467      |
2824       |       0.0        |
|       entrydays_dqx        |  float64   |     31720      |      413      |
44958      |       0.0        |
|       entrydays_dhq        |  float64   |     56337      |     2420      |
20341      |      1071.0      |
|       entrydays_sqx        |  float64   |     49323      |     2972      |
27355      |      3579.0      |
|       entrydays_muq        |  float64   |     28605      |     2142      |
48073      |      6045.0      |
|           center           |   int64    |     76678      |      10       |
0        |       5.0        |
|          rndyear           |   int64    |     76678      |       9       |
0        |      1996.0      |
|            arm             |   int64    |     76678      |       2       |
0        |       1.0        |
|            sex             |   int64    |     76678      |       1       |
0        |       1.0        |
|            age             |   int64    |     76678      |      27       |
0        |       55.0       |
|          agelevel          |   int64    |     76678      |       4       |
0        |       0.0        |
|            dual            |   int64    |     76678      |       2       |
0        |       0.0        |
|     reconsent_outcome      |   int64    |     76678      |      10       |
0        |       1.0        |
|   reconsent_outcome_days   |   int64    |     76678      |     6367      |
0        |      5361.0      |
|      fstcan_exitstat       |   int64    |     76678      |       8       |
0        |       8.0        |
|       fstcan_exitage       |   int64    |     76678      |      36       |
0        |       70.0       |
|      fstcan_exitdays       |   int64    |     76678      |     5878      |
0        |       0.0        |
|    in_TGWAS_population     |   int64    |     76678      |       2       |
0        |       1.0        |
+----------------------------+------------+----------------+---------------+-
--------------+------------------+
```

3.LabelEncoding:

converting catogorical values into numberical

In [88]:

```
from sklearn import preprocessing
def convert(df1):
    number = preprocessing.LabelEncoder()
    df1['plco_id'] = number.fit_transform(df1['plco_id'])
    data = df1.fillna(-9999)
    return data
df_diag1=convert(df1)
```

As there are rows with 99% null values and containg noise data.we took 45 columns without null values

In [2]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
data1=pd.read_csv(r'F:\merged_data_diag_without_null11.csv')
data1

```

...	...	...	...	...	...	...	...
76673	76673	76673	0	0	8	68	Z-162268-9
76674	76674	76674	0	0	8	68	Z-162277-0
76675	76675	76675	0	0	8	69	Z-162295-2
76676	76676	76676	1	1	1	73	Z-162349-7
76677	76677	76677	0	0	5	78	Z-162358-8

76678 rows × 45 columns

In [121]:

```python
from tabulate import tabulate
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, roc_auc_score, precision_score
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

# Assuming you have X_train, y_train, X_test, y_test defined

# Define models
models = {
    'RandomForestClassifier': RandomForestClassifier(max_samples=0.75, ran
    'AdaBoostClassifier(GridSearchCV)': GridSearchCV(estimator=AdaBoostCla
    'SVM classifier':SVC(kernel='linear',random_state=0)
}

# Initialize a list to store results
results = []

# Train and evaluate models
for model_name, model in models.items():
    # Train the model
    model.fit(X_train, y_train)

    # Evaluate on the training set
    y_train_pred = model.predict(X_train)
    train_results = [
        model_name + " (Training)",
        round(accuracy_score(y_train, y_train_pred), 2),
        round(roc_auc_score(y_train, y_train_pred), 2),
        round(precision_score(y_train, y_train_pred), 2),
        round(recall_score(y_train, y_train_pred), 2),
        round(f1_score(y_train, y_train_pred), 2)
    ]

    # Evaluate on the test set
    y_test_pred = model.predict(X_test)
    test_results = [
        model_name + " (Test)",
        round(accuracy_score(y_test, y_test_pred), 2),
        round(roc_auc_score(y_test, y_test_pred), 2),
        round(precision_score(y_test, y_test_pred), 2),
        round(recall_score(y_test, y_test_pred), 2),
        round(f1_score(y_test, y_test_pred), 2)
    ]

    # Append results to the list
    results.extend([train_results, test_results, []])  # Add an empty row

# Display results as a table with left-aligned columns
headers = ["Metric", "Accuracy", "AUC", "Precision", "Recall", "F1 Score"]
print(tabulate(results, headers=headers, tablefmt="pretty", colalign=("lef
```

```
+-------------------------------------------+----------+-----+-----------+-
-------+----------+
| Metric                                    | Accuracy | AUC | Precision |
Recall | F1 Score |
+-------------------------------------------+----------+-----+-----------+-
-------+----------+
| RandomForestClassifier (Training)         | 1.0      | 1.0 | 1.0       |
1.0    | 1.0      |
| RandomForestClassifier (Test)             | 1.0      | 1.0 | 1.0       |
1.0    | 1.0      |
|                                           |          |     |           |
|          |
| AdaBoostClassifier(GridSearchCV) (Training) | 1.0      | 1.0 | 1.0       |
1.0    | 1.0      |
| AdaBoostClassifier(GridSearchCV) (Test)   | 1.0      | 1.0 | 1.0       |
1.0    | 1.0      |
|                                           |          |     |           |
|          |
| SVM classifier (Training)                 | 1.0      | 1.0 | 1.0       |
1.0    | 1.0      |
| SVM classifier (Test)                     | 1.0      | 1.0 | 1.0       |
1.0    | 1.0      |
|                                           |          |     |           |
|          |
+-------------------------------------------+----------+-----+-----------+-
-------+----------+
```

In [3]:
```
from sklearn import preprocessing
def convert(data1):
    number = preprocessing.LabelEncoder()
    data1['plco_id'] = number.fit_transform(data1['plco_id'])
    data1 = data1.fillna(-9999)
    return data1
df_diag1=convert(data1)
```

In [4]:
```
X = data1.drop(columns=['pros_cancer'])
y = data1['pros_cancer']
```

In [12]:
```
import numpy as np
from sklearn.datasets import load_boston
from sklearn.ensemble import RandomForestRegressor,RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFECV
import matplotlib.pyplot as plt
```

In [6]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_
```

In [7]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

# fit the scaler to the train set, it will learn the parameters
scaler.fit(X_train)

# transform train and test sets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [8]:
```
X_train_scaled = pd.DataFrame(X_train_scaled, columns=X_train.columns)
X_test_scaled = pd.DataFrame(X_test_scaled, columns=X_test.columns)
```

In [14]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, classificati
rf = RandomForestClassifier(max_samples=0.75,random_state=42)
rf.fit(X_train_scaled,y_train)
y_pred = rf.predict(X_test_scaled)
accuracy_score(y_test,y_pred)
```

Out[14]:
```
1.0
```

In [15]:
```
rf = RandomForestClassifier(oob_score=True)
rf.fit(X_train,y_train)
```

Out[15]:
```
RandomForestClassifier(oob_score=True)
```

In [16]:
```
rf.oob_score_
```

Out[16]:
```
1.0
```

In [18]:
```
rf.feature_importances_
```

Out[18]:
```
array([4.10091658e-04, 3.58926279e-04, 3.75709531e-01, 3.24406036e-01,
       1.40902291e-02, 4.16303453e-04, 0.00000000e+00, 0.00000000e+00,
       5.41945192e-02, 4.96979168e-03, 1.06592500e-03, 8.06727644e-04,
       2.69560362e-04, 2.21406239e-04, 1.45565550e-03, 3.34888727e-03,
       1.97253956e-03, 1.78469959e-03, 5.50412370e-04, 2.07493364e-04,
       5.42661093e-04, 2.76809976e-04, 2.25156081e-03, 7.26000328e-03,
       3.96728917e-03, 0.00000000e+00, 1.70925218e-02, 3.97417605e-04,
       1.60390969e-04, 3.54424887e-03, 1.24443145e-03, 5.84722343e-04,
       1.66729889e-03, 9.06710143e-05, 0.00000000e+00, 1.02492143e-03,
       6.67641821e-04, 2.08914491e-05, 1.33753753e-02, 1.67707851e-02,
       1.05291712e-01, 5.37313181e-03, 2.96987880e-02, 2.45799135e-03])
```

In [19]:
```
rf.feature_importances_[0]
```

Out[19]:
```
0.00041009165759149174
```

In [24]:
```
1 X.columns
```

Out[24]:
```
Index(['Unnamed: 0.1', 'Unnamed: 0', 'intstatp_cat', 'pros_exitstat',
       'pros_exitage', 'plco_id', 'build_cancers', 'build_incidence_cutoff',
       'pros_exitdays', 'biopplink0', 'biopplink1', 'biopplink2', 'biopplink
3',
       'biopplink4', 'biopplink5', 'pros_mra_stat0', 'pros_mra_stat1',
       'pros_mra_stat2', 'pros_mra_stat3', 'pros_mra_stat4', 'pros_mra_stat
5',
       'race7', 'is_dead', 'mortality_exitage', 'mortality_exitstat',
       'build_death_cutoff', 'mortality_exitdays', 'ph_any_trial',
       'ph_pros_trial', 'pros_eligible_bq', 'pros_eligible_dhq', 'center',
       'rndyear', 'arm', 'sex', 'age', 'agelevel', 'dual', 'reconsent_outcom
e',
       'reconsent_outcome_days', 'fstcan_exitstat', 'fstcan_exitage',
       'fstcan_exitdays', 'in_TGWAS_population'],
      dtype='object')
```

In [25]:
```
1 for column in X.columns,feature in rf.feature_importances_:
2     print(feature,column)
```

```
0.00041009165759149174 Index(['Unnamed: 0.1', 'Unnamed: 0', 'intstatp_cat',
'pros_exitstat',
       'pros_exitage', 'plco_id', 'build_cancers', 'build_incidence_cutoff',
       'pros_exitdays', 'biopplink0', 'biopplink1', 'biopplink2', 'biopplink
3',
       'biopplink4', 'biopplink5', 'pros_mra_stat0', 'pros_mra_stat1',
       'pros_mra_stat2', 'pros_mra_stat3', 'pros_mra_stat4', 'pros_mra_stat
5',
       'race7', 'is_dead', 'mortality_exitage', 'mortality_exitstat',
       'build_death_cutoff', 'mortality_exitdays', 'ph_any_trial',
       'ph_pros_trial', 'pros_eligible_bq', 'pros_eligible_dhq', 'center',
       'rndyear', 'arm', 'sex', 'age', 'agelevel', 'dual', 'reconsent_outcom
e',
       'reconsent_outcome_days', 'fstcan_exitstat', 'fstcan_exitage',
       'fstcan_exitdays', 'in_TGWAS_population'],
      dtype='object')
0.00041009165759149174 True
```

In [26]:

```
for feature in rf.feature_importances_:
    print(feature,column)
```

```
0.00041009165759149174 True
0.000358926279083804 True
0.37570953061722606 True
0.32440603568931076 True
0.014090229091711353 True
0.00041630345330046127 True
0.0 True
0.0 True
0.054194519205205814 True
0.00496979167822959 True
0.0010659249971144767 True
0.0008067276439515991 True
0.00026956036183300324 True
0.00022140623946249695 True
0.0014556554974967698 True
0.003348887268018343 True
0.001972539561810602 True
0.0017846995882557296 True
0.0005504123702712078 True
0.00020749336448690515 True
0.0005426610927860889 True
0.00027680997641196136 True
0.0022515608065287692 True
0.007260003282694108 True
0.003967289173486258 True
0.0 True
0.01709252177716915 True
0.00039741760496309287 True
0.0001603909692532981 True
0.0035442488684008745 True
0.0012444314511888168 True
0.0005847223429693237 True
0.001667298886849651 True
9.067101426305714e-05 True
0.0 True
0.0010249214292640647 True
0.0006676418214172717 True
2.089144914201813e-05 True
0.013375375314348563 True
0.016770785113458343 True
0.1052917119392919 True
0.005373131809764228 True
0.02969878796248623 True
0.0024579913495024752 True
```

In [27]:
```
zipped=zip(X.columns,rf.feature_importances_)
list(zipped)
```

Out[27]:
```
[('Unnamed: 0.1', 0.00041009165759149174),
 ('Unnamed: 0', 0.000358926279083804),
 ('intstatp_cat', 0.37570953061722606),
 ('pros_exitstat', 0.32440603568931076),
 ('pros_exitage', 0.014090229091711353),
 ('plco_id', 0.00041630345330046127),
 ('build_cancers', 0.0),
 ('build_incidence_cutoff', 0.0),
 ('pros_exitdays', 0.054194519205205814),
 ('biopplink0', 0.00496979167822959),
 ('biopplink1', 0.0010659249971144767),
 ('biopplink2', 0.0008067276439515991),
 ('biopplink3', 0.00026956036183300324),
 ('biopplink4', 0.00022140623946249695),
 ('biopplink5', 0.0014556554974967698),
 ('pros_mra_stat0', 0.003348887268018343),
 ('pros_mra_stat1', 0.001972539561810602),
 ('pros_mra_stat2', 0.0017846995882557296),
 ('pros_mra_stat3', 0.0005504123702712078),
 ('pros_mra_stat4', 0.00020749336448690515),
 ('pros_mra_stat5', 0.0005426610927860889),
 ('race7', 0.00027680997641196136),
 ('is_dead', 0.0022515608065287692),
 ('mortality_exitage', 0.007260003282694108),
 ('mortality_exitstat', 0.003967289173486258),
 ('build_death_cutoff', 0.0),
 ('mortality_exitdays', 0.01709252177716915),
 ('ph_any_trial', 0.00039741760496309287),
 ('ph_pros_trial', 0.0001603909692532981),
 ('pros_eligible_bq', 0.0035442488684008745),
 ('pros_eligible_dhq', 0.0012444314511888168),
 ('center', 0.0005847223429693237),
 ('rndyear', 0.001667298886849651),
 ('arm', 9.067101426305714e-05),
 ('sex', 0.0),
 ('age', 0.0010249214292640647),
 ('agelevel', 0.0006676418214172717),
 ('dual', 2.089144914201813e-05),
 ('reconsent_outcome', 0.013375375314348563),
 ('reconsent_outcome_days', 0.016770785113458343),
 ('fstcan_exitstat', 0.1052917119392919),
 ('fstcan_exitage', 0.005373131809764228),
 ('fstcan_exitdays', 0.02969878796248623),
 ('in_TGWAS_population', 0.0024579913495024752)]
```

Here we applied Randomforest for feature selection

set1

In [94]:
```
X=data1.iloc[:,2:21]
Y=data1['pros_cancer']
```

In [95]:
```
X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size=0.2,random_
```

In [96]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data1['plco_id'] = le.fit_transform(data1['plco_id'])
```

In [97]:
```
# Number of trees in random forest
n_estimators = [20,60,100,120]

# Number of features to consider at every split
max_features = [0.2,0.6,1.0]

# Maximum number of levels in tree
max_depth = [2,8,None]

# Number of samples
max_samples = [0.5,0.75,1.0]

# 108 diff random forest train
```

In [98]:
```
param_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
              'max_samples':max_samples
             }
print(param_grid)
```

```
{'n_estimators': [20, 60, 100, 120], 'max_features': [0.2, 0.6, 1.0], 'max_de
pth': [2, 8, None], 'max_samples': [0.5, 0.75, 1.0]}
```

In [99]:
```
rf = RandomForestClassifier(max_samples=0.75,random_state=42)
rf.fit(X_train,y_train)
y_pred = rf.predict(X_test)
accuracy_score(y_test,y_pred)
```

Out[99]: 1.0

In [100]:
```
from sklearn.metrics import classification_report
print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     13563
           1       1.00      1.00      1.00      1773

    accuracy                           1.00     15336
   macro avg       1.00      1.00      1.00     15336
weighted avg       1.00      1.00      1.00     15336
```

set2

In [101]:
```
X=data1.iloc[:,21:43]
Y=data1['pros_cancer']
```

In [102]:
```
X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size=0.2,random_
param_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
              'max_samples':max_samples
             }
print(param_grid)
# Number of trees in random forest
n_estimators = [20,60,100,120]

# Number of features to consider at every split
max_features = [0.2,0.6,1.0]

# Maximum number of levels in tree
max_depth = [2,8,None]

# Number of samples
max_samples = [0.5,0.75,1.0]

# 108 diff random forest train
```

```
{'n_estimators': [20, 60, 100, 120], 'max_features': [0.2, 0.6, 1.0], 'max_de
pth': [2, 8, None], 'max_samples': [0.5, 0.75, 1.0]}
```

In [103]:
```
rf = RandomForestClassifier(max_samples=0.75,random_state=42)
rf.fit(X_train,y_train)
y_pred = rf.predict(X_test)
accuracy_score(y_test,y_pred)
```

Out[103]:
```
0.9334898278560251
```

In [104]:
```
from sklearn.metrics import classification_report
print(classification_report(y_pred, y_test))
```

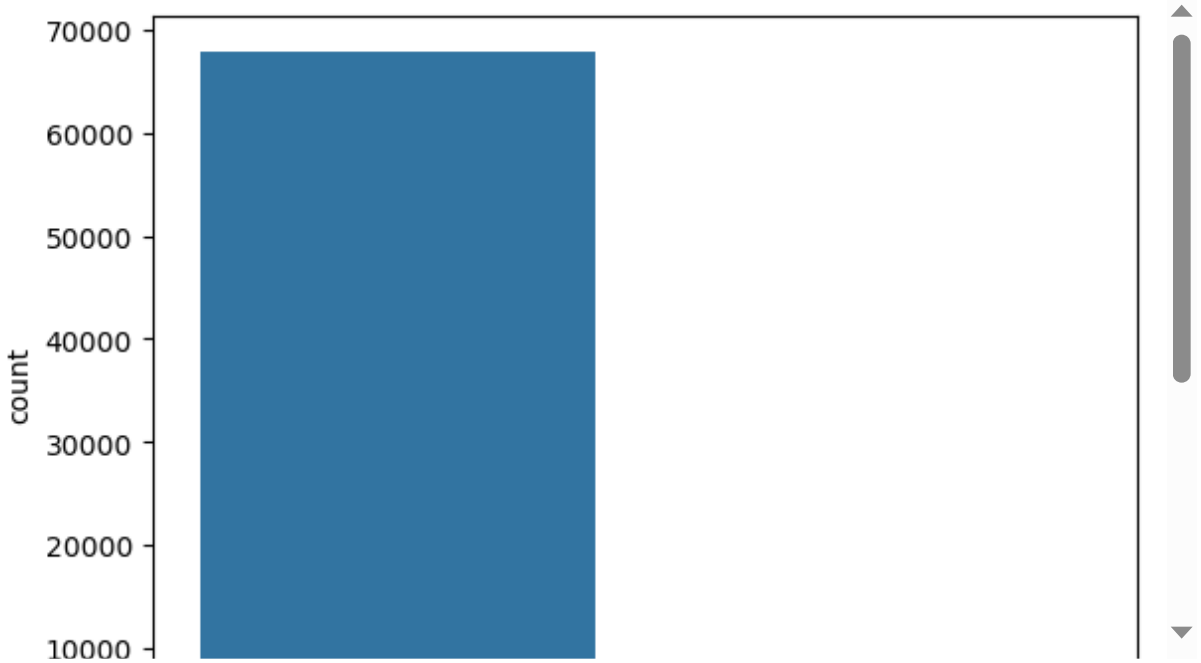
```
              precision    recall  f1-score   support

           0       0.96      0.97      0.96     13453
           1       0.74      0.70      0.72      1883

    accuracy                           0.93     15336
   macro avg       0.85      0.83      0.84     15336
weighted avg       0.93      0.93      0.93     15336
```

In [11]:

```
# Exploring the variable'diagnosis'
sns.countplot(x='pros_cancer', data=data1, label='Count')
plt.show()
print("\nNumber of no cases (labeled as 0)")
print((data1.pros_cancer  == 0).sum())
print("\nNumber of yes cases (labeled as 1):")
print((data1.pros_cancer  == 1).sum())
```



#As we getting 0 accuracy due noise present in the dataset so we making subsets of the data to select the features to imporve accurany.features selected and main dataset are from 21 to 45 colums

here we got accurancy of 0.93 for feature from 21 to 45

In [40]:
```
o1=pd.read_csv(r"O1.csv")
o1
```

Out[40]:

	Unnamed: 0	pros_mra_stat5	race7	is_dead	mortality_exitage	mortality_exitstat	build_d
0	0	0	2	1	88	1	
1	1	0	1	0	81	2	
2	2	0	1	0	83	2	
3	3	0	1	1	70	1	
4	4	0	1	1	73	1	

In [46]:
```
X1 = o1.drop(columns=['pros_cancer'])
y1 = o1['pros_cancer']
```

In [10]:
```
X1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76678 entries, 0 to 76677
Data columns (total 22 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   pros_mra_stat5         76678 non-null  int64
 1   race7                  76678 non-null  int64
 2   is_dead                76678 non-null  int64
 3   mortality_exitage      76678 non-null  int64
 4   mortality_exitstat     76678 non-null  int64
 5   build_death_cutoff     76678 non-null  int64
 6   mortality_exitdays     76678 non-null  int64
 7   ph_any_trial           76678 non-null  int64
 8   ph_pros_trial          76678 non-null  int64
 9   pros_eligible_bq       76678 non-null  int64
 10  pros_eligible_dhq      76678 non-null  int64
 11  center                 76678 non-null  int64
 12  rndyear                76678 non-null  int64
 13  arm                    76678 non-null  int64
```

In [42]:
```
df=o1.drop(columns=['plco_id','Unnamed: 0'], inplace=True)
```

In [43]:
```
1 o1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76678 entries, 0 to 76677
Data columns (total 23 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   pros_mra_stat5          76678 non-null  int64
 1   race7                   76678 non-null  int64
 2   is_dead                 76678 non-null  int64
 3   mortality_exitage       76678 non-null  int64
 4   mortality_exitstat      76678 non-null  int64
 5   build_death_cutoff      76678 non-null  int64
 6   mortality_exitdays      76678 non-null  int64
 7   ph_any_trial            76678 non-null  int64
 8   ph_pros_trial           76678 non-null  int64
 9   pros_eligible_bq        76678 non-null  int64
 10  pros_eligible_dhq       76678 non-null  int64
 11  center                  76678 non-null  int64
 12  rndyear                 76678 non-null  int64
 13  arm                     76678 non-null  int64
```

In [ ]:
```
1 o1
```

In [47]:
```
1 X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.3,
```

In [56]:

```
from tabulate import tabulate
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, roc_auc_score, precision_score
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

# Assuming you have X_train, y_train, X_test, y_test defined

# Define models
models = {
    'RandomForestClassifier': RandomForestClassifier(max_samples=0.75, ran
    'AdaBoostClassifier(GridSearchCV)': GridSearchCV(estimator=AdaBoostCla
    'SVM classifier':SVC(kernel='linear',random_state=0)
}

# Initialize a list to store results
results = []

# Train and evaluate models
for model_name, model in models.items():
    # Train the model
    model.fit(X_train, y_train)

    # Evaluate on the training set
    y_train_pred = model.predict(X_train)
    train_results = [
        model_name + " (Training)",
        round(accuracy_score(y_train, y_train_pred), 2),
        round(roc_auc_score(y_train, y_train_pred), 2),
        round(precision_score(y_train, y_train_pred), 2),
        round(recall_score(y_train, y_train_pred), 2),
        round(f1_score(y_train, y_train_pred), 2)
    ]

    # Evaluate on the test set
    y_test_pred = model.predict(X_test)
    test_results = [
        model_name + " (Test)",
        round(accuracy_score(y_test, y_test_pred), 2),
        round(roc_auc_score(y_test, y_test_pred), 2),
        round(precision_score(y_test, y_test_pred), 2),
        round(recall_score(y_test, y_test_pred), 2),
        round(f1_score(y_test, y_test_pred), 2)
    ]

    # Append results to the list
    results.extend([train_results, test_results, []])  # Add an empty row

# Display results as a table with left-aligned columns
headers = ["Metric", "Accuracy", "AUC", "Precision", "Recall", "F1 Score"]
print(tabulate(results, headers=headers, tablefmt="pretty", colalign=("lef
```

Metric	Accuracy	AUC	Precision	Recall	F1 Score
RandomForestClassifier (Training)	1.0	1.0	1.0	1.0	1.0
RandomForestClassifier (Test)	0.93	0.86	0.69	0.75	0.72
AdaBoostClassifier(GridSearchCV) (Training)	0.94	0.86	0.71	0.76	0.74
AdaBoostClassifier(GridSearchCV) (Test)	0.93	0.85	0.7	0.74	0.72
SVM classifier (Training)	0.93	0.86	0.66	0.78	0.71
SVM classifier (Test)	0.93	0.85	0.65	0.76	0.7

In [13]:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classificati
from sklearn.metrics import precision_recall_fscore_support

# Assuming X is your feature matrix and y is the target variable
# Make sure your target variable (y) is binary (0 or 1)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.2,

# Initialize the RandomForest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = rf_classifier.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

# Calculate confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Extract true positives, true negatives, false positives, and false negat
tn, fp, fn, tp = conf_matrix.ravel()

# Calculate sensitivity (true positive rate)
sensitivity = tp / (tp + fn)

# Calculate specificity (true negative rate)
specificity = tn / (tn + fp)

# Calculate precision, recall, and F1-score
precision, recall, f1_score, _ = precision_recall_fscore_support(y_test, y

# Display the results
print(f"Accuracy: {accuracy:.2f}")
print(f"Sensitivity: {sensitivity:.2f}")
print(f"Specificity: {specificity:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1_score:.2f}")

# Additional classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

```

Accuracy: 0.93
Sensitivity: 0.74
Specificity: 0.96
Precision: 0.69
Recall: 0.74
F1 Score: 0.72

Classification Report:

	precision	recall	f1-score	support
0	0.97	0.96	0.96	13563
1	0.69	0.74	0.72	1773
accuracy			0.93	15336
macro avg	0.83	0.85	0.84	15336
weighted avg	0.93	0.93	0.93	15336

In [20]:

```
from sklearn.model_selection import cross_val_predict, cross_val_score, KF
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import make_scorer, precision_recall_fscore_support

# Assuming X is your feature matrix and y is the target variable
# Make sure your target variable (y) is binary (0 or 1)

# Initialize the RandomForest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

# Define the number of folds for k-fold cross-validation
n_folds = 5

# Create a KFold object
kf = KFold(n_splits=n_folds, shuffle=True, random_state=42)

# Specify the metrics you want to use for evaluation
scoring = {'accuracy': 'accuracy',
           'precision': 'precision',
           'recall': 'recall',
           'f1_score': 'f1'}

# Perform k-fold cross-validation
cross_val_results = cross_val_score(rf_classifier, X1, y1, cv=kf, scoring=

# Display the results
print(f'Cross-Validation Accuracy: {cross_val_results.mean():.2f}')

# Additional metrics
for metric in ['precision', 'recall', 'f1_score']:
    cross_val_results_metric = cross_val_predict(rf_classifier, X1, y1, cv
    scores = precision_recall_fscore_support(y1, (cross_val_results_metric
    print(f'Cross-Validation {metric.capitalize()}: {scores[0 if metric=="
```

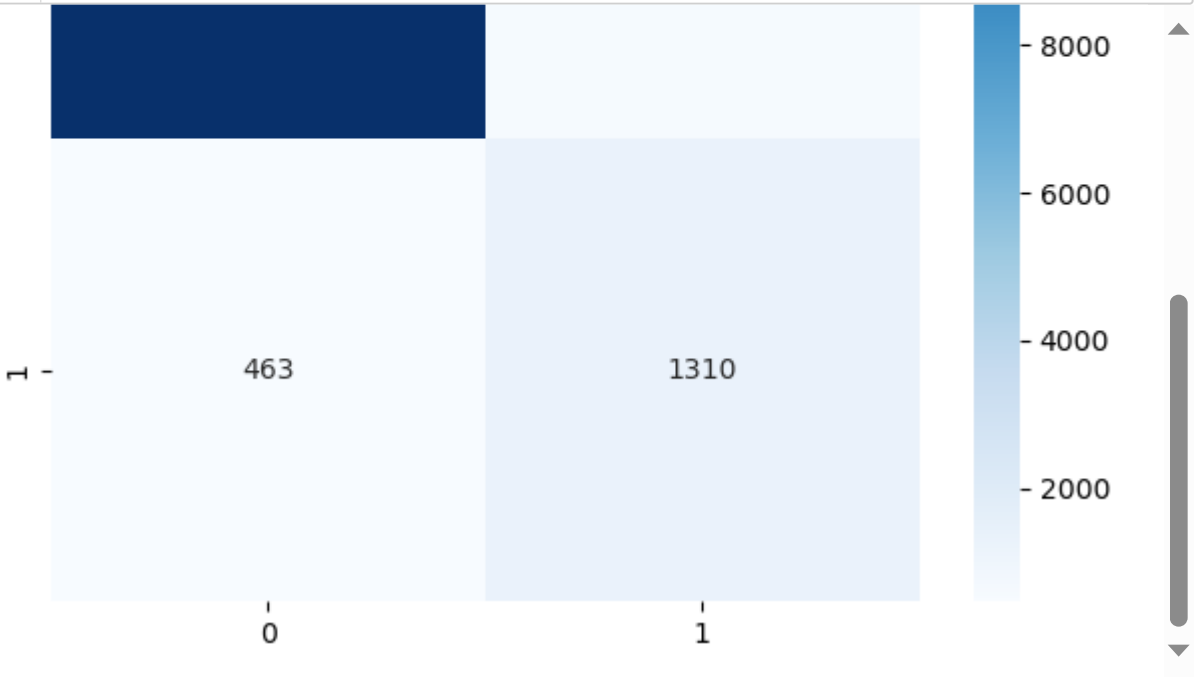
```
Cross-Validation Accuracy: 0.93
Cross-Validation Precision: 0.68
Cross-Validation Recall: 0.74
Cross-Validation F1_score: 0.74
```

In [27]:

```
rf_classifier=rf_classifier.predict(X_test)
accuracy=accuracy_score(y_test,rf_classifier )
```

In [34]:

```
cm = confusion_matrix(y_test,y_pred_class)
print(cm)
plt.figure(figsize=(7, 6))
sns.heatmap(confusion_matrix(y_test,rf_classifier), annot=True, fmt='d', c
plt.title('Confusion Matrix for Randomforest')
plt.show()
```



Cross validation

In [42]:

```
cv_score_lr = cross_val_score(RandomForestClassifier(n_estimators=100, ran

print(cv_score_lr)

mean_accuracy_lr = sum(cv_score_lr)/len(cv_score_lr)

mean_accuracy_lr = mean_accuracy_lr*100

mean_accuracy_lr = round(mean_accuracy_lr, 2)

print(mean_accuracy_lr)
```

```
[0.93009911 0.93218571 0.9331638  0.9291164  0.93152918]
93.12
```

In [48]:

```
from sklearn.model_selection import cross_val_predict, cross_val_score, KF
cv_score_svc = cross_val_score(SVC(kernel='linear'), X1, y1, cv=5)

print(cv_score_svc)

mean_accuracy_svc = sum(cv_score_svc)/len(cv_score_svc)

mean_accuracy_svc = mean_accuracy_svc*100

mean_accuracy_svc = round(mean_accuracy_svc, 2)

print(mean_accuracy_svc)
```

```
[0.92788211 0.92846896 0.93068597 0.92813825 0.92729051]
92.85
```