```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
import sklearn
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import RepeatedKFold, cross_val_score

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

```
df=pd.read_csv('dataset_med.csv')
```

```
df.head()
```

	id	age	gender	country	diagnosis_date	cancer_stage	family_history
0	1	64.0	Male	Sweden	2016-04-05	Stage I	Yes
1	2	50.0	Female	Netherlands	2023-04-20	Stage III	Yes
2	3	65.0	Female	Hungary	2023-04-05	Stage III	Yes
3	4	51.0	Female	Belgium	2016-02-05	Stage I	No
4	5	37.0	Male	Luxembourg	2023-11-29	Stage I	No

```
df.shape
```

```
(80733, 17)
```

```
df.columns
```

```
Index(['id', 'age', 'gender', 'country', 'diagnosis_date', 'cancer_stage',
       'family_history', 'smoking_status', 'bmi', 'cholesterol_level',
       'hypertension', 'asthma', 'cirrhosis', 'other_cancer',
'treatment_type',
       'end_treatment_date', 'survived'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 80733 entries, 0 to 80732
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   id                  80733 non-null  int64  
 1   age                 80733 non-null  float64
 2   gender              80733 non-null  object 
 3   country             80733 non-null  object 
 4   diagnosis_date      80733 non-null  object 
 5   cancer_stage        80733 non-null  object 
 6   family_history      80732 non-null  object 
 7   smoking_status      80732 non-null  object 
 8   bmi                 80732 non-null  float64
 9   cholesterol_level   80732 non-null  float64
 10  hypertension        80732 non-null  float64
 11  asthma              80732 non-null  float64
 12  cirrhosis           80732 non-null  float64
 13  other_cancer        80732 non-null  float64
 14  treatment_type      80732 non-null  object 
 15  end_treatment_date  80732 non-null  object 
 16  survived            80732 non-null  float64
dtypes: float64(8), int64(1), object(8)
memory usage: 10.5+ MB
```

```
df = df.drop('id', axis=1)
```

```
print(df.duplicated().sum())
```

```
0
```

```
df.describe()
```

	age	bmi	cholesterol_level	hypertension	asthma
count	80733.000000	80732.000000	80732.000000	80732.000000	80732.000000
mean	54.961651	30.496871	233.716791	0.752415	0.466655
std	9.981384	8.376907	43.495299	0.431612	0.498890
min	15.000000	16.000000	150.000000	0.000000	0.000000
25%	48.000000	23.200000	197.000000	1.000000	0.000000
50%	55.000000	30.500000	242.000000	1.000000	0.000000
75%	62.000000	37.800000	271.000000	1.000000	1.000000
max	101.000000	45.000000	300.000000	1.000000	1.000000

```
binary_col = ["hypertension", "asthma", "cirrhosis", "other_cancer", "survived'
category_col = ["cancer_stage", "smoking_status", "treatment_type"]

# Fill missing values in binary columns with 0
df[binary_col] = df[binary_col].fillna(0)

# Fill missing values in numerical columns with the mean
numerical_cols_to_fill = ['age', 'bmi', 'cholesterol_level']
df[numerical_cols_to_fill] = df[numerical_cols_to_fill].fillna(df[numerical_col

# Fill missing values in categorical columns with a placeholder
df[category_col] = df[category_col].fillna('Unknown')

df['age'] = df['age'].astype("int8")
df[binary_col] = df[binary_col].astype("int8")
df["gender"] = df["gender"].map({'Female': 1, 'Male': 0}).fillna(0).astype("int
df["family_history"] = df["family_history"].map({'Yes': 1, 'No': 0}).fillna(0).
df[category_col] = df[category_col].astype("category")

# Convert date columns to datetime objects and calculate treatment_time
df['diagnosis_date'] = pd.to_datetime(df['diagnosis_date'], errors='coerce')
df['end_treatment_date'] = pd.to_datetime(df['end_treatment_date'], errors='coe
df['treatment_time'] = (df['end_treatment_date'] - df['diagnosis_date']).dt.day

# Fill NaN/NaT values in treatment_time with a placeholder (e.g., -1) before cc
df['treatment_time'] = df['treatment_time'].fillna(-1).astype(int)



# Drop the original date columns
```

```
df = df.drop(['diagnosis_date','end_treatment_date'],axis=1)


df.info()
df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 80733 entries, 0 to 80732
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   age                 80733 non-null  int8
 1   gender              80733 non-null  int8
 2   country             80733 non-null  object
 3   cancer_stage        80733 non-null  category
 4   family_history      80733 non-null  int8
 5   smoking_status      80733 non-null  category
 6   bmi                 80733 non-null  float64
 7   cholesterol_level   80733 non-null  float64
 8   hypertension        80733 non-null  int8
 9   asthma              80733 non-null  int8
 10  cirrhosis           80733 non-null  int8
 11  other_cancer        80733 non-null  int8
 12  treatment_type      80733 non-null  category
 13  survived            80733 non-null  int8
 14  treatment_time      80733 non-null  int64
dtypes: category(3), float64(2), int64(1), int8(8), object(1)
memory usage: 3.3+ MB
```

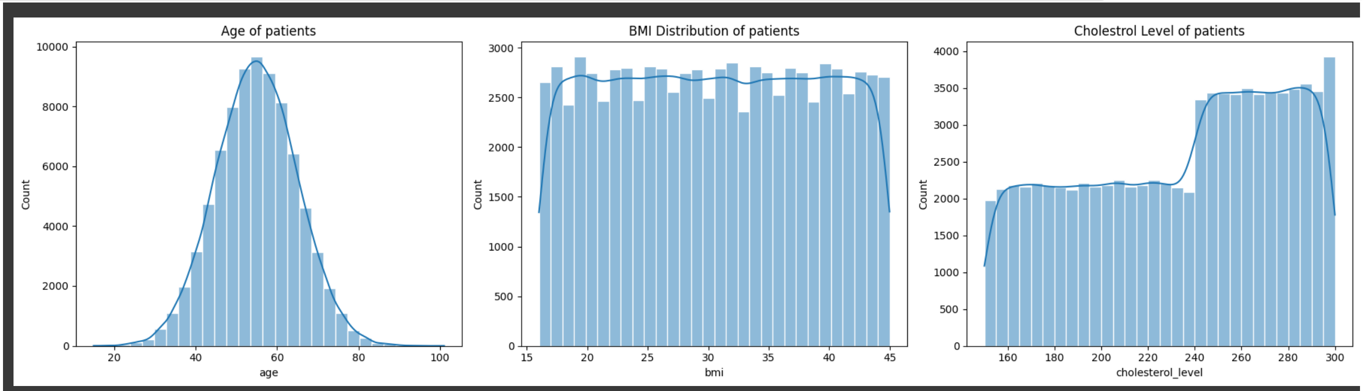
	age	gender	country	cancer_stage	family_history	smoking_status	bmi
0	64	0	Sweden	Stage I	1	Passive Smoker	29.4
1	50	1	Netherlands	Stage III	1	Passive Smoker	41.2
2	65	1	Hungary	Stage III	1	Former Smoker	44.0
3	51	1	Belgium	Stage I	0	Passive Smoker	43.0
4	37	0	Luxembourg	Stage I	0	Passive Smoker	19.7

```
from sklearn.preprocessing import LabelEncoder
import pandas as pd

le = LabelEncoder()
df['cancer_stage'] = le.fit_transform(df['cancer_stage'])

df = pd.get_dummies(df, columns=['smoking_status', 'treatment_type'], drop_firs

df.head()
```

	age	gender	country	cancer_stage	family_history	bmi	cholesterol_le
0	64	0	Sweden	1	1	29.4	19
1	50	1	Netherlands	3	1	41.2	28
2	65	1	Hungary	3	1	44.0	26
3	51	1	Belgium	1	0	43.0	24
4	37	0	Luxembourg	1	0	19.7	17

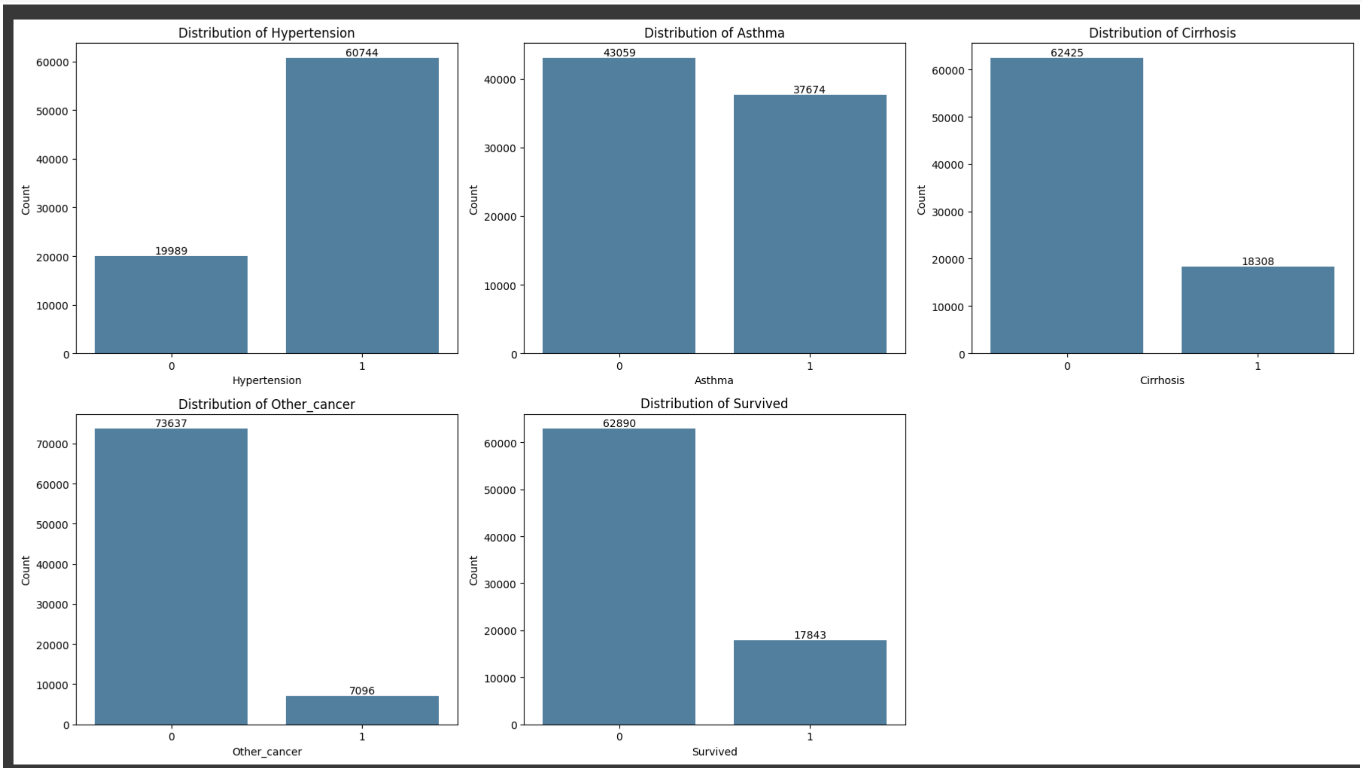
5 rows × 21 columns

```
fig, axes = plt.subplots(1, 3, figsize=(18, 5))

sns.histplot(df, x="age", binwidth=3, edgecolor="white", element="bars", kde=Tr
axes[0].set_title("Age of patients")

sns.histplot(df["bmi"], bins=30, edgecolor="white", kde=True, ax=axes[1])
axes[1].set_title("BMI Distribution of patients")

sns.histplot(df["cholesterol_level"], bins=30, edgecolor="white", kde=True, ax=
axes[2].set_title("Cholestrol Level of patients")

plt.tight_layout()
fig.show()
```

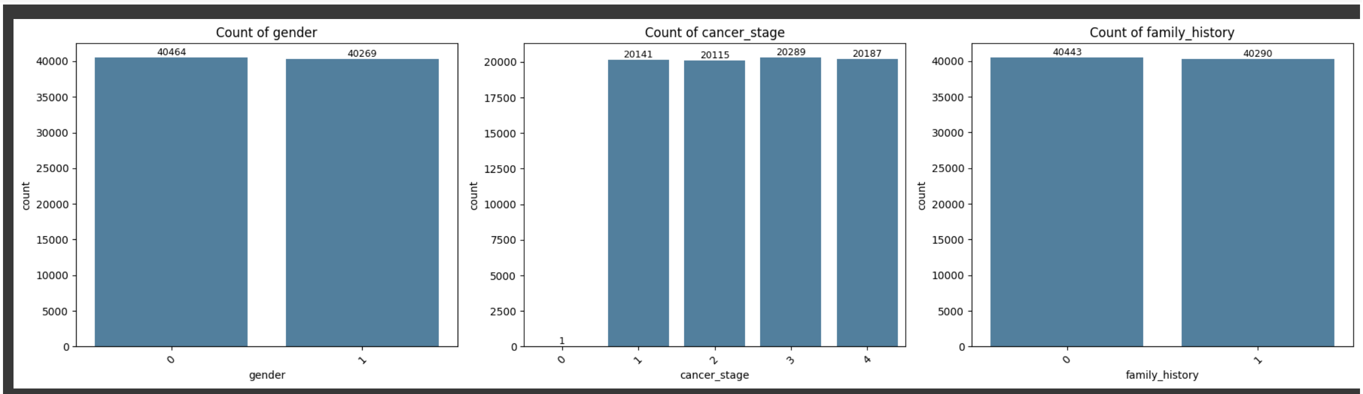


```
cols = ["hypertension", "asthma", "cirrhosis", "other_cancer", "survived"]

fig, axes = plt.subplots(2, 3, figsize=(18, 10))

axes = axes.flatten()

for i, col in enumerate(cols):
    sns.countplot(x=col, data=df, ax=axes[i], color="#4682A9")
    for p in axes[i].patches:
        height = p.get_height()
        axes[i].annotate(f'{int(height)}',
                         (p.get_x() + p.get_width() / 2., height),
                         ha='center', va='bottom')
    axes[i].set_title(f"Distribution of {col.capitalize()}")
    axes[i].set_xlabel(col.capitalize())
    axes[i].set_ylabel("Count")
```

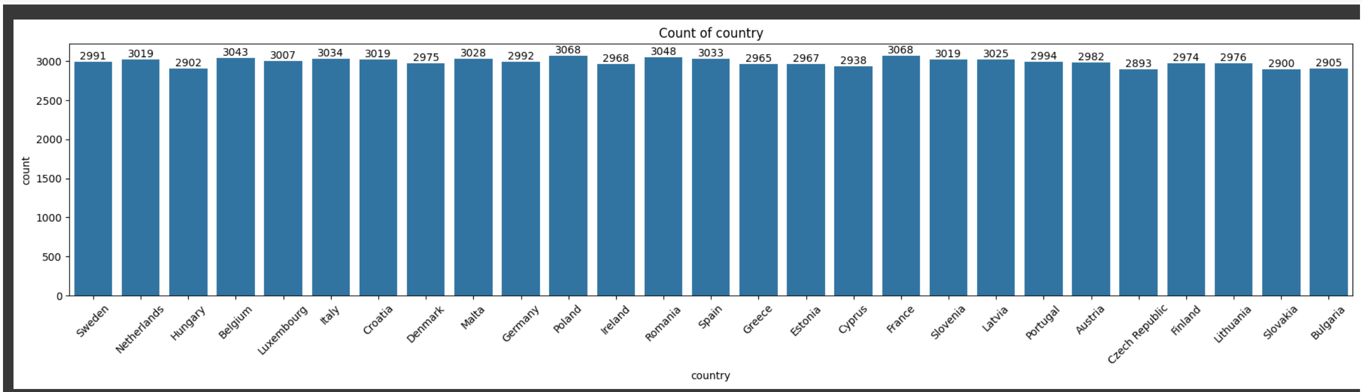
```
if len(cols) < len(axes):
    fig.delaxes(axes[-1])  # Remove unused subplot

plt.tight_layout()
plt.show()
```



```
category = ['gender', 'cancer_stage',
            'family_history']

fig, axes = plt.subplots(1, 3, figsize=(18, 5))

for ax, col in zip(axes.flatten(), category):
    sns.countplot(data=df, x=col, ax=ax, color="#4682A9")

    for p in ax.patches:
        height = p.get_height()
        ax.annotate(f'{int(height)}',
                    (p.get_x() + p.get_width() / 2, height),
                    ha='center', va='bottom', fontsize=9)

    ax.set_title(f'Count of {col}')
    ax.tick_params(axis='x', rotation=45)

axes_flat = axes.flatten()
if len(category) < len(axes_flat):
    for i in range(len(category), len(axes_flat)):
        fig.delaxes(axes_flat[i])

plt.tight_layout()
plt.show()
```



```
plt.figure(figsize=(18, 5))
ax = sns.countplot(data=df, x="country")
plt.title("Count of country")
plt.xticks(rotation=45)

# Add labels on top of bars
for p in ax.patches:
    height = p.get_height()
    ax.annotate(f'{int(height)}',
                (p.get_x() + p.get_width() / 2, height),
                ha='center', va='bottom')

plt.tight_layout()
plt.show()
```



```
df_model=df.copy()
```

```
scaler = StandardScaler()
diff_unit = ["age", "bmi", "cholesterol_level", "treatment_time"]
df_model[diff_unit] = scaler.fit_transform(df_model[diff_unit])

df_model.head()
```

	age	gender	country	cancer_stage	family_history	bmi	choles
0	0.905526	0	Sweden	1	1	-0.130941	
1	-0.497094	1	Netherlands	3	1	1.277710	
2	1.005713	1	Hungary	3	1	1.611967	
3	-0.396906	1	Belgium	1	0	1.492590	
4	-1.799526	0	Luxembourg	1	0	-1.288901	

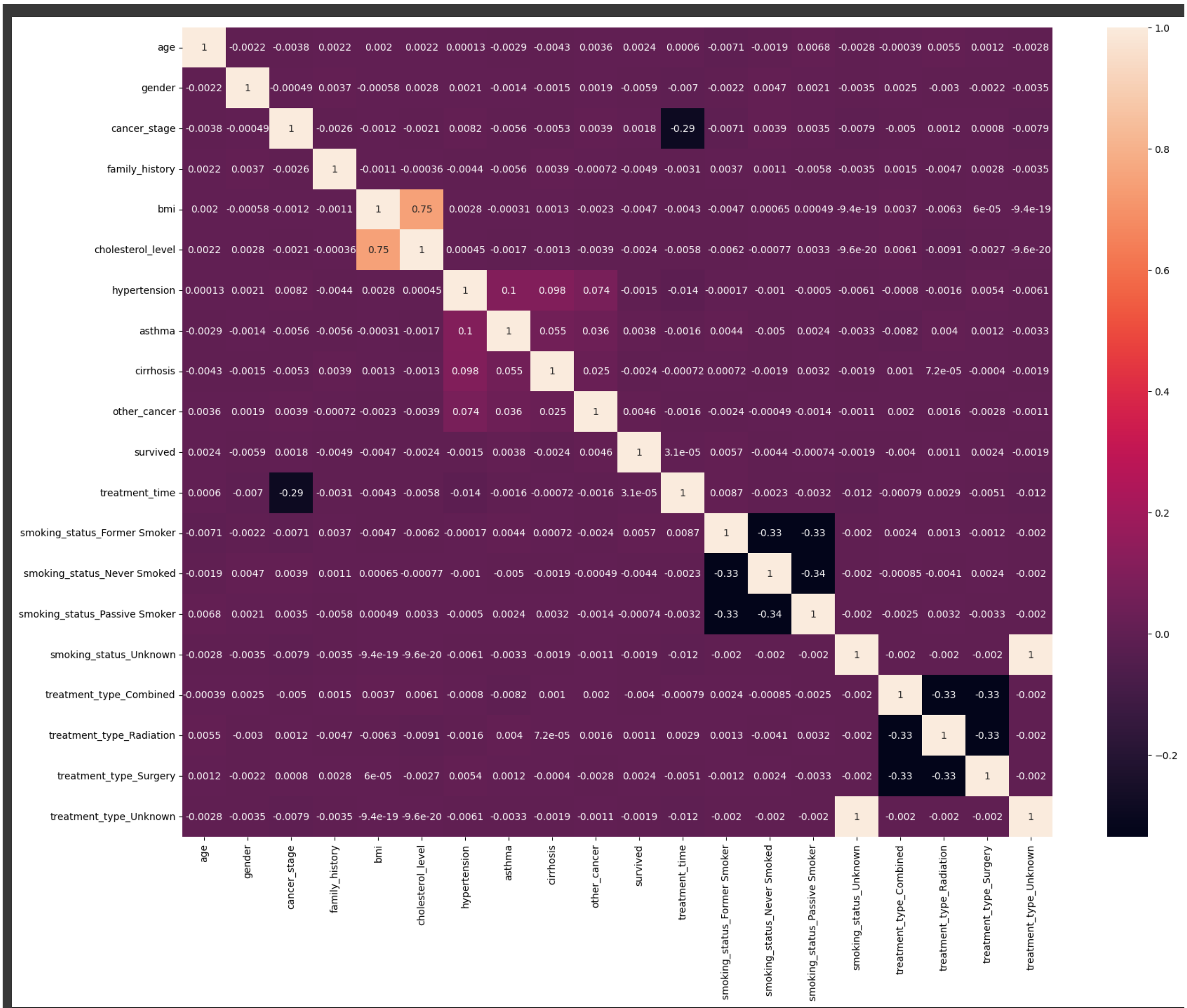
5 rows × 21 columns

```
corr = df_model.corr(numeric_only=True)
print(corr)
```

```
                                 age    gender  cancer_stage  \
age                         1.000000 -0.002221     -0.003801   
gender                     -0.002221  1.000000     -0.000489   
cancer_stage               -0.003801 -0.000489      1.000000   
family_history              0.002229  0.003749     -0.002629   
bmi                         0.002007 -0.000582     -0.001202   
cholesterol_level           0.002242  0.002760     -0.002119   
hypertension                0.000134  0.002087      0.008178   
asthma                     -0.002869 -0.001366     -0.005568   
cirrhosis                  -0.004315 -0.001532     -0.005262   
other_cancer                0.003551  0.001887      0.003934   
survived                    0.002426 -0.005908      0.001805   
treatment_time              0.000596 -0.006978     -0.289022   
smoking_status_Former Smoker  -0.007115 -0.002232     -0.007114   
smoking_status_Never Smoked   -0.001931  0.004712      0.003873   
smoking_status_Passive Smoker  0.006795  0.002114      0.003497   
smoking_status_Unknown     -0.002807 -0.003511     -0.007879   
treatment_type_Combined    -0.000393  0.002451     -0.004990   
treatment_type_Radiation    0.005471 -0.003039      0.001206   
treatment_type_Surgery      0.001156 -0.002190      0.000802   
treatment_type_Unknown     -0.002807 -0.003511     -0.007879   

                           family_history           bmi  \
age                              0.002229  2.007202e-03   
gender                           0.003749 -5.820157e-04   
cancer_stage                    -0.002629 -1.201590e-03   
```

```
family_history                       1.000000 -1.065642e-03
bmi                                 -0.001066  1.000000e+00
cholesterol_level                   -0.000358  7.480522e-01
hypertension                        -0.004445  2.758397e-03
asthma                              -0.005626 -3.059462e-04
cirrhosis                            0.003925  1.329084e-03
other_cancer                        -0.000724 -2.268058e-03
survived                            -0.004931 -4.734229e-03
treatment_time                      -0.003118 -4.295175e-03
smoking_status_Former Smoker         0.003670 -4.723461e-03
smoking_status_Never Smoked          0.001097  6.496740e-04
smoking_status_Passive Smoker       -0.005782  4.918737e-04
smoking_status_Unknown              -0.003513 -9.354595e-19
treatment_type_Combined              0.001464  3.749931e-03
treatment_type_Radiation            -0.004656 -6.339500e-03
treatment_type_Surgery               0.002839  6.018004e-05
treatment_type_Unknown              -0.003513 -9.354595e-19

                               cholesterol_level  hypertension    asthma
age                                 2.242015e-03      0.000134 -0.002869
gender                              2.760413e-03      0.002087 -0.001366
cancer_stage                       -2.119179e-03      0.008178 -0.005568
family_history                     -3.579847e-04     -0.004445 -0.005626
bmi                                 7.480522e-01      0.002758 -0.000306
cholesterol_level                   1.000000e+00      0.000446 -0.001701
hypertension                        4.455175e-04      1.000000  0.103190
asthma                             -1.700743e-03      0.103190  1.000000
cirrhosis                          -1.288772e-03      0.097863  0.055177
other_cancer                       -3.940041e-03      0.074388  0.036185
survived                           -2.410907e-03     -0.001534  0.003804
treatment_time                     -5.801593e-03     -0.013976 -0.001586
smoking_status_Former Smoker       -6.157001e-03     -0.000170  0.004439
```

```
plt.figure(figsize=(20,15))
sns.heatmap(corr, annot=True)
plt.show()
```



	age	gender	cancer_stage	family_history	bmi	cholesterol_level	hypertension	asthma	cirrhosis	other_cancer	survived	treatment_time	smoking_status_Former Smoker	smoking_status_Never Smoked	smoking_status_Passive Smoker	smoking_status_Unknown	treatment_type_Combined	treatment_type_Radiation	treatment_type_Surgery	treatment_type_Unknown
age	1	-0.0022	-0.0038	0.0022	0.002	0.0022	0.00013	-0.0029	-0.0043	0.0036	0.0024	0.0006	-0.0071	-0.0019	0.0068	-0.0028	-0.00039	0.0055	0.0012	-0.0028
gender	-0.0022	1	-0.00049	0.0037	-0.00058	0.0028	0.0021	-0.0014	-0.0015	0.0019	-0.0059	-0.007	-0.0022	0.0047	0.0021	-0.0035	0.0025	-0.003	-0.0022	-0.0035
cancer_stage	-0.0038	-0.00049	1	-0.0026	-0.0012	-0.0021	0.0082	-0.0056	-0.0053	0.0039	0.0018	-0.29	-0.0071	0.0039	0.0035	-0.0079	-0.005	0.0012	0.0008	-0.0079
family_history	0.0022	0.0037	-0.0026	1	-0.0011	-0.00036	-0.0044	-0.0056	0.0039	-0.00072	-0.0049	-0.0031	0.0037	0.0011	-0.0058	-0.0035	0.0015	-0.0047	0.0028	-0.0035
bmi	0.002	-0.00058	-0.0012	-0.0011	1	0.75	0.0028	-0.00031	0.0013	-0.0023	-0.0047	-0.0043	-0.0047	0.00065	0.00049	-9.4e-19	0.0037	-0.0063	6e-05	-9.4e-19
cholesterol_level	0.0022	0.0028	-0.0021	-0.00036	0.75	1	0.00045	-0.0017	-0.0013	-0.0039	-0.0024	-0.0058	-0.0062	-0.00077	0.0033	-9.6e-20	0.0061	-0.0091	-0.0027	-9.6e-20
hypertension	0.00013	0.0021	0.0082	-0.0044	0.0028	0.00045	1	0.1	0.098	0.074	-0.0015	-0.014	-0.00017	-0.001	-0.0005	-0.0061	-0.0008	-0.0016	0.0054	-0.0061
asthma	-0.0029	-0.0014	-0.0056	-0.0056	-0.00031	-0.0017	0.1	1	0.055	0.036	0.0038	-0.0016	0.0044	-0.005	0.0024	-0.0033	-0.0082	0.004	0.0012	-0.0033
cirrhosis	-0.0043	-0.0015	-0.0053	0.0039	0.0013	-0.0013	0.098	0.055	1	0.025	-0.0024	-0.00072	0.00072	-0.0019	0.0032	-0.0019	0.001	7.2e-05	-0.0004	-0.0019
other_cancer	0.0036	0.0019	0.0039	-0.00072	-0.0023	-0.0039	0.074	0.036	0.025	1	0.0046	-0.0016	-0.0024	-0.00049	-0.0014	-0.0011	0.002	0.0016	-0.0028	-0.0011
survived	0.0024	-0.0059	0.0018	-0.0049	-0.0047	-0.0024	-0.0015	0.0038	-0.0024	0.0046	1	3.1e-05	0.0057	-0.0044	-0.00074	-0.0019	-0.004	0.0011	0.0024	-0.0019
treatment_time	0.0006	-0.007	-0.29	-0.0031	-0.0043	-0.0058	-0.014	-0.0016	-0.00072	-0.0016	3.1e-05	1	0.0087	-0.0023	-0.0032	-0.012	-0.00079	0.0029	-0.0051	-0.012
smoking_status_Former Smoker	-0.0071	-0.0022	-0.0071	0.0037	-0.0047	-0.0062	-0.00017	0.0044	0.00072	-0.0024	0.0057	0.0087	1	-0.33	-0.33	-0.002	0.0024	0.0013	-0.0012	-0.002
smoking_status_Never Smoked	-0.0019	0.0047	0.0039	0.0011	0.00065	-0.00077	-0.001	-0.005	-0.0019	-0.00049	-0.0044	-0.0023	-0.33	1	-0.34	-0.002	-0.00085	-0.0041	0.0024	-0.002
smoking_status_Passive Smoker	0.0068	0.0021	0.0035	-0.0058	0.00049	0.0033	-0.0005	0.0024	0.0032	-0.0014	-0.00074	-0.0032	-0.33	-0.34	1	-0.002	-0.0025	0.0032	-0.0033	-0.002
smoking_status_Unknown	-0.0028	-0.0035	-0.0079	-0.0035	-9.4e-19	-9.6e-20	-0.0061	-0.0033	-0.0019	-0.0011	-0.0019	-0.012	-0.002	-0.002	-0.002	1	-0.002	-0.002	-0.002	1
treatment_type_Combined	-0.00039	0.0025	-0.005	0.0015	0.0037	0.0061	-0.0008	-0.0082	0.001	0.002	-0.004	-0.00079	0.0024	-0.00085	-0.0025	-0.002	1	-0.33	-0.33	-0.002
treatment_type_Radiation	0.0055	-0.003	0.0012	-0.0047	-0.0063	-0.0091	-0.0016	0.004	7.2e-05	0.0016	0.0011	0.0029	0.0013	-0.0041	0.0032	-0.002	-0.33	1	-0.33	-0.002
treatment_type_Surgery	0.0012	-0.0022	0.0008	0.0028	6e-05	-0.0027	0.0054	0.0012	-0.0004	-0.0028	0.0024	-0.0051	-0.0012	0.0024	-0.0033	-0.002	-0.33	-0.33	1	-0.002
treatment_type_Unknown	-0.0028	-0.0035	-0.0079	-0.0035	-9.4e-19	-9.6e-20	-0.0061	-0.0033	-0.0019	-0.0011	-0.0019	-0.012	-0.002	-0.002	-0.002	1	-0.002	-0.002	-0.002	1

```
category = ['gender', 'country', 'cancer_stage',
       'family_history']

for i in category:
  print(f"{i.capitalize()}\nUnique values: {df_model[i].unique()}\nCount of uni
```

[redacted]

```
 'Denmark' 'Malta' 'Germany' 'Poland' 'Ireland' 'Romania' 'Spain' 'Greece'
 'Estonia' 'Cyprus' 'France' 'Slovenia' 'Latvia' 'Portugal' 'Austria'
 'Czech Republic' 'Finland' 'Lithuania' 'Slovakia' 'Bulgaria']
Count of unique values: 27

Cancer_stage
Unique values: [1 3 4 2 0]
Count of unique values: 5

Family_history
Unique values: [1 0]
Count of unique values: 2
```

```
one_hot_columns = [
    'gender',
    'family_history',
]

df_encoded = pd.get_dummies(df_model[one_hot_columns], drop_first=False)

df_encoded = df_encoded.replace({True: 1, False: 0}).astype(int)
```

```
from sklearn.model_selection import train_test_split

# Define features (X) and target (y)
X = df_model.drop('survived', axis=1)
y = df_model['survived']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random

print("Training set shape:", X_train.shape)
print("Testing set shape:", X_test.shape)
```

```
Training set shape: (64586, 20)
Testing set shape: (16147, 20)
```

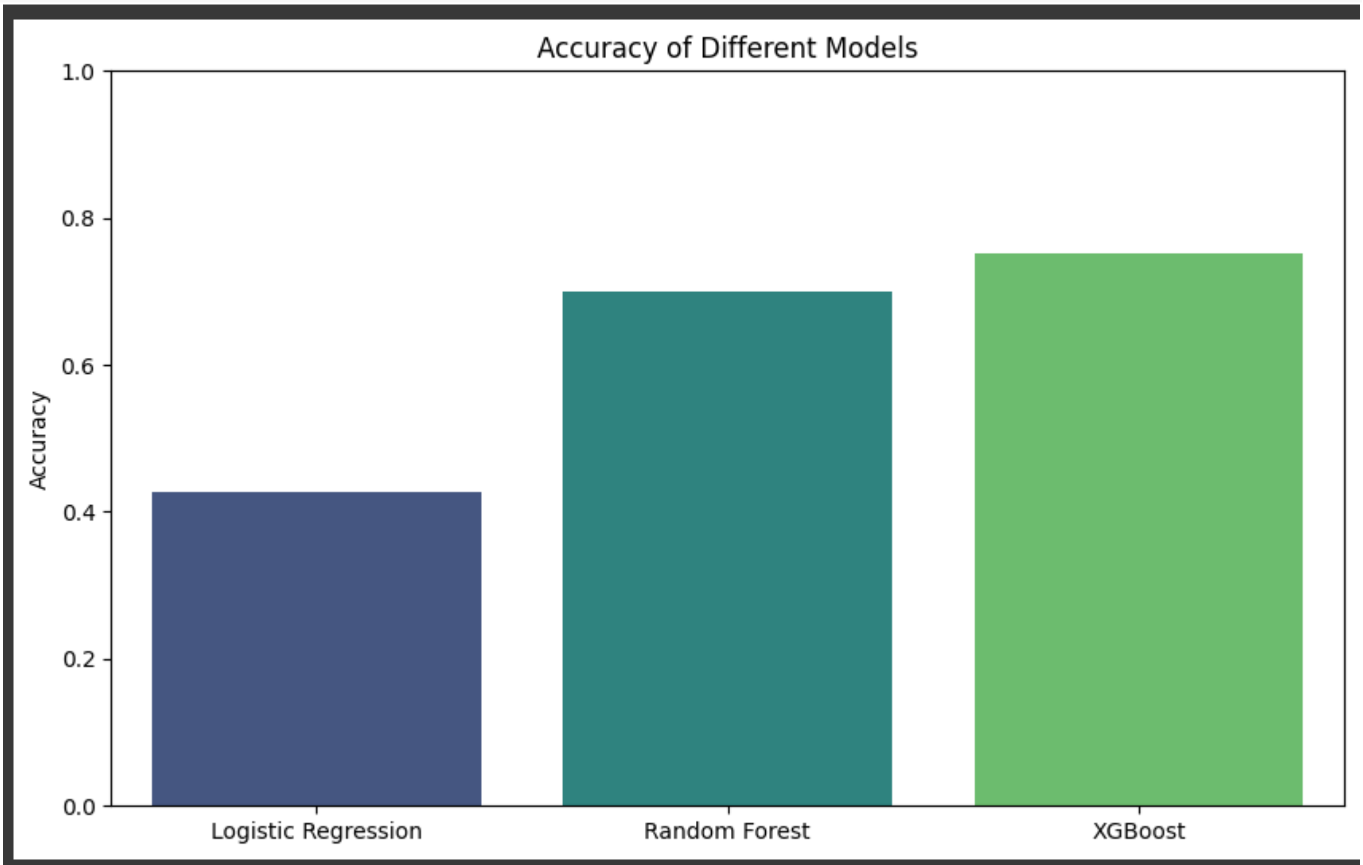
```
from imblearn.over_sampling import SMOTE

# Apply SMOTE only to the training data
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train.select_dtypes(include

print("Shape of training data after SMOTE:", X_train_smote.shape)
print("Distribution of 'survived' in training data after SMOTE:")
print(y_train_smote.value_counts())
```

```
Shape of training data after SMOTE: (100624, 11)
Distribution of 'survived' in training data after SMOTE:
survived
0    50312
1    50312
Name: count, dtype: int64
```

```
# Initialize models
log_reg = LogisticRegression(max_iter=1000, random_state=42)
rf_clf = RandomForestClassifier(random_state=42)
xgb_clf = XGBClassifier(random_state=42)

models = {
    "Logistic Regression": log_reg,
    "Random Forest": rf_clf,
    "XGBoost": xgb_clf
}

accuracy_scores = {}

# Train and evaluate models
for name, model in models.items():
    print(f"Training {name}...")
    model.fit(X_train_smote, y_train_smote)
    y_pred = model.predict(X_test.select_dtypes(include=np.number))
    report = classification_report(y_test, y_pred, output_dict=True)
    accuracy = report['accuracy']
    accuracy_scores[name] = accuracy
    print(f"Evaluation for {name}:")
    print(classification_report(y_test, y_pred))
    print("-" * 30)
```

```
Training Logistic Regression...
Evaluation for Logistic Regression:
              precision    recall  f1-score   support

           0       0.77      0.38      0.51     12578
           1       0.21      0.60      0.32      3569

    accuracy                           0.43     16147
   macro avg       0.49      0.49      0.41     16147
weighted avg       0.65      0.43      0.46     16147

------------------------------
Training Random Forest...
Evaluation for Random Forest:
              precision    recall  f1-score   support

           0       0.78      0.85      0.82     12578
           1       0.23      0.16      0.19      3569

    accuracy                           0.70     16147
   macro avg       0.51      0.51      0.50     16147
weighted avg       0.66      0.70      0.68     16147

------------------------------
Training XGBoost...
Evaluation for XGBoost:
              precision    recall  f1-score   support

           0       0.78      0.95      0.86     12578
           1       0.24      0.06      0.09      3569

    accuracy                           0.75     16147
   macro avg       0.51      0.50      0.47     16147
weighted avg       0.66      0.75      0.69     16147

------------------------------
```

```
# Plot the accuracy of the models
plt.figure(figsize=(10, 6))
sns.barplot(x=list(accuracy_scores.keys()), y=list(accuracy_scores.values()), pa
plt.title("Accuracy of Different Models")
plt.ylabel("Accuracy")
plt.ylim(0, 1)
plt.show()
```


Accuracy of Different Models
Accuracy
Logistic Regression
Random Forest
XGBoost