```
import pandas as pd, numpy as np, matplotlib.pyplot as plt, warnings, os, joblib
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.metrics import (accuracy_score, precision_score, recall_score, f1_score,
                             roc_auc_score, classification_report, confusion_matrix, roc_curve, auc)
warnings.filterwarnings('ignore')
plt.rcParams['figure.figsize'] = (8,5)
```

```
data=pd.read_csv('/content/Lung_Cancer.csv')
df=pd.DataFrame(data)
print('Loaded dataset with shape:',df.shape)
df.head()
```

```
Loaded dataset with shape: (890000, 17)
```

	id	age	gender	country	diagnosis_date	cancer_stage	family_history	smoking_status	bmi	cholesterol_level	hypert
0	1	64	Male	Sweden	05-04-2016	Stage I	Yes	Passive Smoker	29.4	199	
1	2	50	Female	Netherlands	20-04-2023	Stage III	Yes	Passive Smoker	41.2	280	
2	3	65	Female	Hungary	05-04-2023	Stage III	Yes	Former Smoker	44.0	268	
3	4	51	Female	Belgium	05-02-2016	Stage I	No	Passive Smoker	43.0	241	
4	5	37	Male	Luxembourg	29-11-2023	Stage I	No	Passive Smoker	19.7	178	

```
print('Dtypes:')
df.dtypes
```

```
Dtypes:
```

	0
id	int64
age	int64
gender	object
country	object
diagnosis_date	object
cancer_stage	object
family_history	object
smoking_status	object
bmi	float64
cholesterol_level	int64
hypertension	int64
asthma	int64
cirrhosis	int64
other_cancer	int64
treatment_type	object
end_treatment_date	object
survived	int64

dtype: object

```
print('\nMissing values:')
print(df.isnull().sum().sort_values(ascending=False).head(30))
```

```
Missing values:
id                   0
age                  0
gender               0
country              0
diagnosis_date       0
cancer_stage         0
```

```
family_history        0
smoking_status        0
bmi                   0
cholesterol_level     0
hypertension          0
asthma                0
cirrhosis             0
other_cancer          0
treatment_type        0
end_treatment_date    0
survived              0
dtype: int64
```

```
display(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 890000 entries, 0 to 889999
Data columns (total 17 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   id                  890000 non-null  int64  
 1   age                 890000 non-null  int64  
 2   gender              890000 non-null  object 
 3   country             890000 non-null  object 
 4   diagnosis_date      890000 non-null  object 
 5   cancer_stage        890000 non-null  object 
 6   family_history      890000 non-null  object 
 7   smoking_status      890000 non-null  object 
 8   bmi                 890000 non-null  float64
 9   cholesterol_level   890000 non-null  int64  
 10  hypertension        890000 non-null  int64  
 11  asthma              890000 non-null  int64  
 12  cirrhosis           890000 non-null  int64  
 13  other_cancer        890000 non-null  int64  
 14  treatment_type      890000 non-null  object 
 15  end_treatment_date  890000 non-null  object 
 16  survived            890000 non-null  int64  
dtypes: float64(1), int64(8), object(8)
memory usage: 115.4+ MB
None
```

```
display(df.describe(include='all').T)
```

	count	unique	top	freq	mean	std	min	25%	50%	75%	
id	890000.0	NaN	NaN	NaN	445000.5	256921.014128	1.0	222500.75	445000.5	667500.25	89
age	890000.0	NaN	NaN	NaN	55.007008	9.994485	4.0	48.0	55.0	62.0	
gender	890000	2	Male	445134	NaN	NaN	NaN	NaN	NaN	NaN	
country	890000	27	Malta	33367	NaN	NaN	NaN	NaN	NaN	NaN	
diagnosis_date	890000	3651	15-05-2024	306	NaN	NaN	NaN	NaN	NaN	NaN	
cancer_stage	890000	4	Stage III	222594	NaN	NaN	NaN	NaN	NaN	NaN	
family_history	890000	2	No	445181	NaN	NaN	NaN	NaN	NaN	NaN	
smoking_status	890000	4	Passive Smoker	223170	NaN	NaN	NaN	NaN	NaN	NaN	
bmi	890000.0	NaN	NaN	NaN	30.494172	8.368539	16.0	23.3	30.5	37.7	
cholesterol_level	890000.0	NaN	NaN	NaN	233.633916	43.432278	150.0	196.0	242.0	271.0	
hypertension	890000.0	NaN	NaN	NaN	0.750024	0.432999	0.0	1.0	1.0	1.0	
asthma	890000.0	NaN	NaN	NaN	0.46974	0.499084	0.0	0.0	0.0	1.0	
cirrhosis	890000.0	NaN	NaN	NaN	0.225956	0.418211	0.0	0.0	0.0	0.0	
other_cancer	890000.0	NaN	NaN	NaN	0.088157	0.283524	0.0	0.0	0.0	0.0	
treatment_type	890000	4	Chemotherapy	223262	NaN	NaN	NaN	NaN	NaN	NaN	
end_treatment_date	890000	4194	09-12-2023	294	NaN	NaN	NaN	NaN	NaN	NaN	

```
dups = df.duplicated().sum()
print('\nNumber of duplicate rows:', dups)
```

```

Number of duplicate rows: 0
```

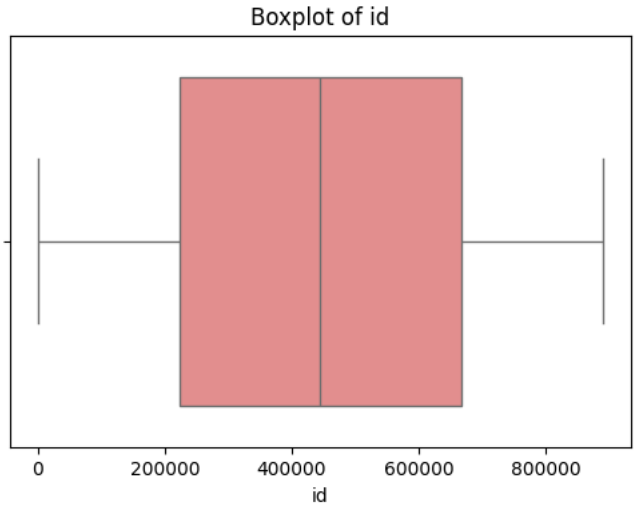
```
df['survived'].value_counts()
```

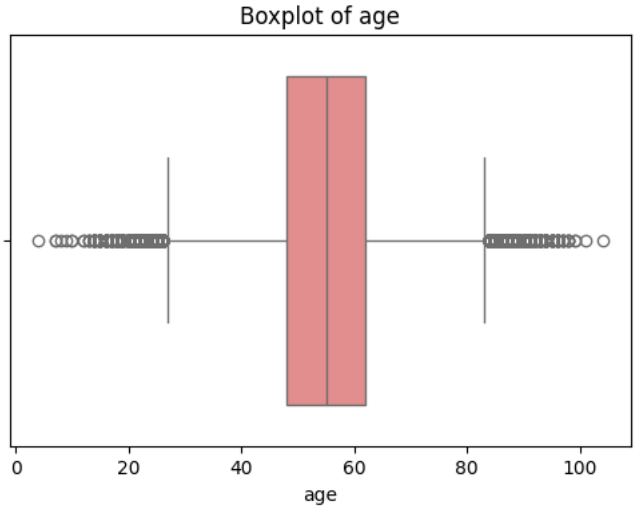
survived	count
0	693996
1	196004

dtype: int64

```
num_cols = df.select_dtypes(include=['int64', 'float64']).columns.tolist()
```

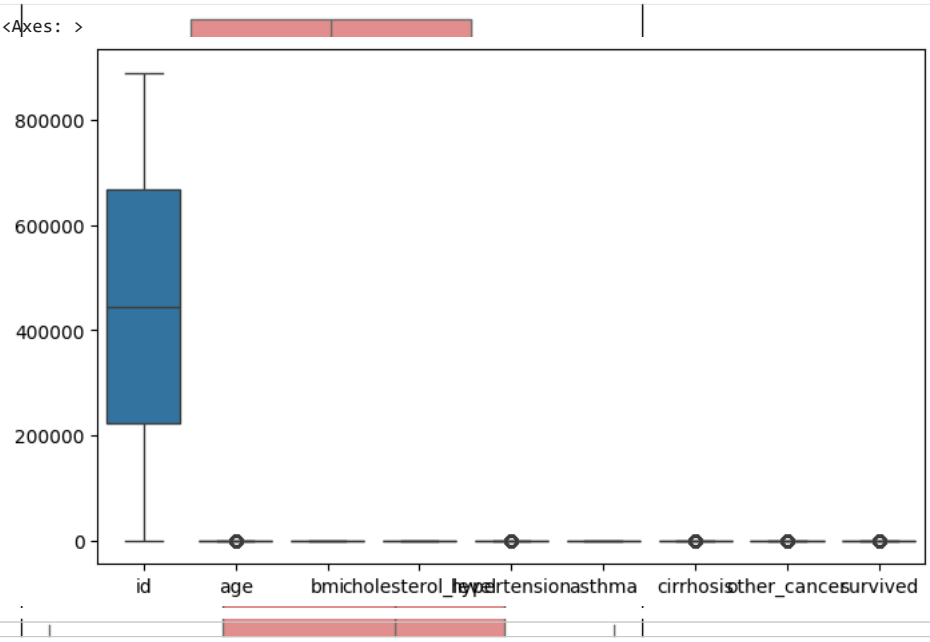
```
import seaborn as sns
top_num = num_cols[:6] if len(num_cols) > 6 else num_cols
for col in top_num:
    plt.figure(figsize=(6,4))
    sns.boxplot(x=df[col], color='lightcoral')
    plt.title(f'Boxplot of {col}')
    plt.show
```






```
sns.boxplot(data=df)
```

```
<Axes: >
```




```
def handling_outliers(df, exclude_cols=None):
    if exclude_cols is None:
        exclude_cols = []

    for col in df.select_dtypes(include='number').columns:
        if col in exclude_cols:
            continue  # skip target or excluded columns

        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - (1.5 * IQR)
        upper_bound = Q3 + (1.5 * IQR)
```

```
        # Capping outliers
        df[col] = df[col].apply(lambda x: lower_bound if x < lower_bound
                                else upper_bound if x > upper_bound else x)
    return df
```

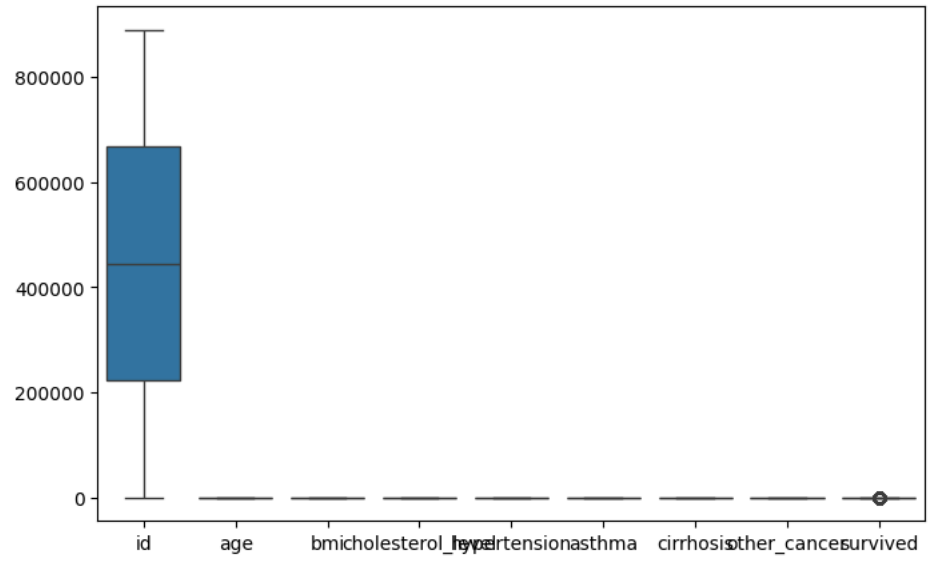
```
df1 = handling_outliers(df, exclude_cols=['survived'])
df1
```

	id	age	gender	country	diagnosis_date	cancer_stage	family_history	smoking_status	bmi	cholesterol_lev
0	1	64.0	Male	Sweden	05-04-2016	Stage I	Yes	Passive Smoker	29.4	1
1	2	50.0	Female	Netherlands	20-04-2023	Stage III	Yes	Passive Smoker	41.2	2
2	3	65.0	Female	Hungary	05-04-2023	Stage III	Yes	Former Smoker	44.0	2
3	4	51.0	Female	Belgium	05-02-2016	Stage I	No	Passive Smoker	43.0	2
4	5	37.0	Male	Luxembourg	29-11-2023	Stage I	No	Passive Smoker	19.7	1
					hypertension					
...	...	...	...	...	...	...	...	...	...	
889995	889996	40.0	Male	Malta	01-07-2022	Stage IV	No	Passive Smoker	44.8	2
889996	889997	62.0	Female	Cyprus	27-09-2015	Stage II	Yes	Former Smoker	21.6	2
889997	889998	48.0	Female	Estonia	27-03-2016	Stage III	Yes	Never Smoked	38.6	2
889998	889999	67.0	Female	Slovakia	22-12-2015	Stage IV	Yes	Former Smoker	18.6	1
889999	890000	55.0	Female	Malta	26-07-2021	Stage II	Yes	Current Smoker	42.8	2

890000 rows × 17 columns

```
sns.boxplot(data=df1)
```

```
<Axes: >
```




```
df1['survived'].value_counts()
```

survived	count
0	693996
1	196004

dtype: int64

```
from sklearn.preprocessing import LabelEncoder
df_encoded = df.copy()
categorical_cols = df_encoded.select_dtypes(include=['object', 'category']).columns.tolist()
le = LabelEncoder()
for col in categorical_cols:
    df_encoded[col] = le.fit_transform(df_encoded[col].astype(str))

print("Label encoding complete!")
```

```
print("Encoded DataFrame shape:", df_encoded.shape)
df_encoded.head()
```

```
Label encoding complete!
Encoded DataFrame shape: (890000, 17)
```

	id	age	gender	country	diagnosis_date	cancer_stage	family_history	smoking_status	bmi	cholesterol_level	hypertens
0	1	64.0	1	26	510	0	1	3	29.4	199	
1	2	50.0	0	19	2317	2	1	3	41.2	280	
2	3	65.0	0	12	517	2	1	1	44.0	268	
3	4	51.0	0	1	490	0	0	3	43.0	241	
4	5	37.0	1	17	3461	0	0	3	19.7	178	

```
from imblearn.over_sampling import SMOTE

X = df_encoded.drop(columns=['survived'])
y = df_encoded['survived']

smote = SMOTE(random_state=42)
X_balanced, y_balanced = smote.fit_resample(X, y)

print(y_balanced.value_counts())
```

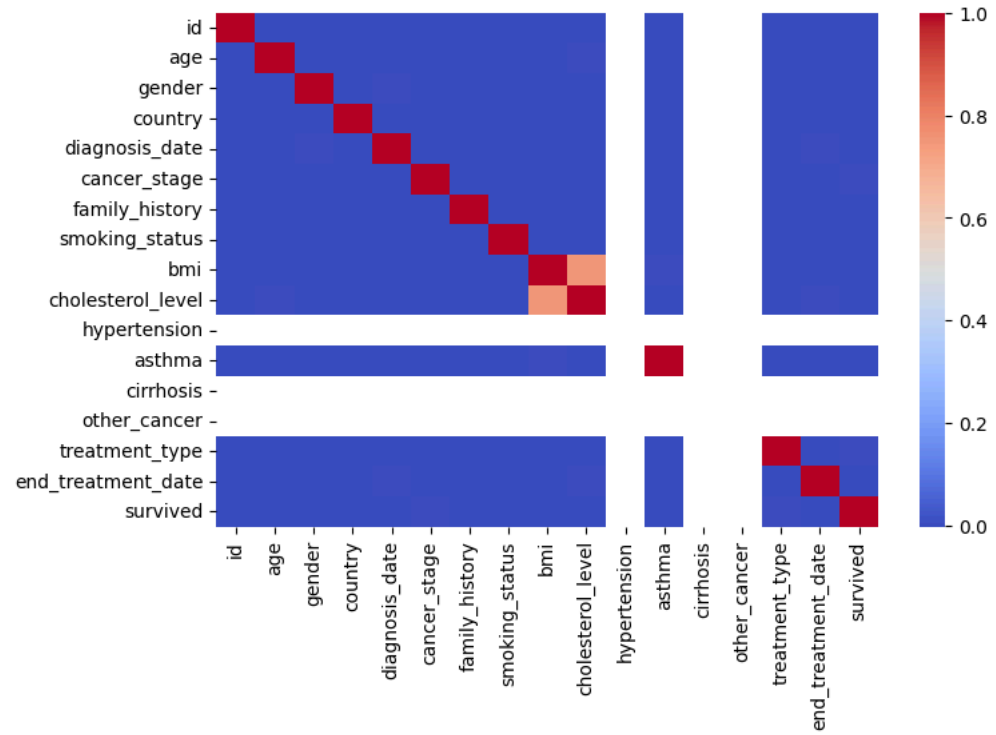
```
survived
0    693996
1    693996
Name: count, dtype: int64
```

```
c=df_encoded.corr(numeric_only=True)
c
```

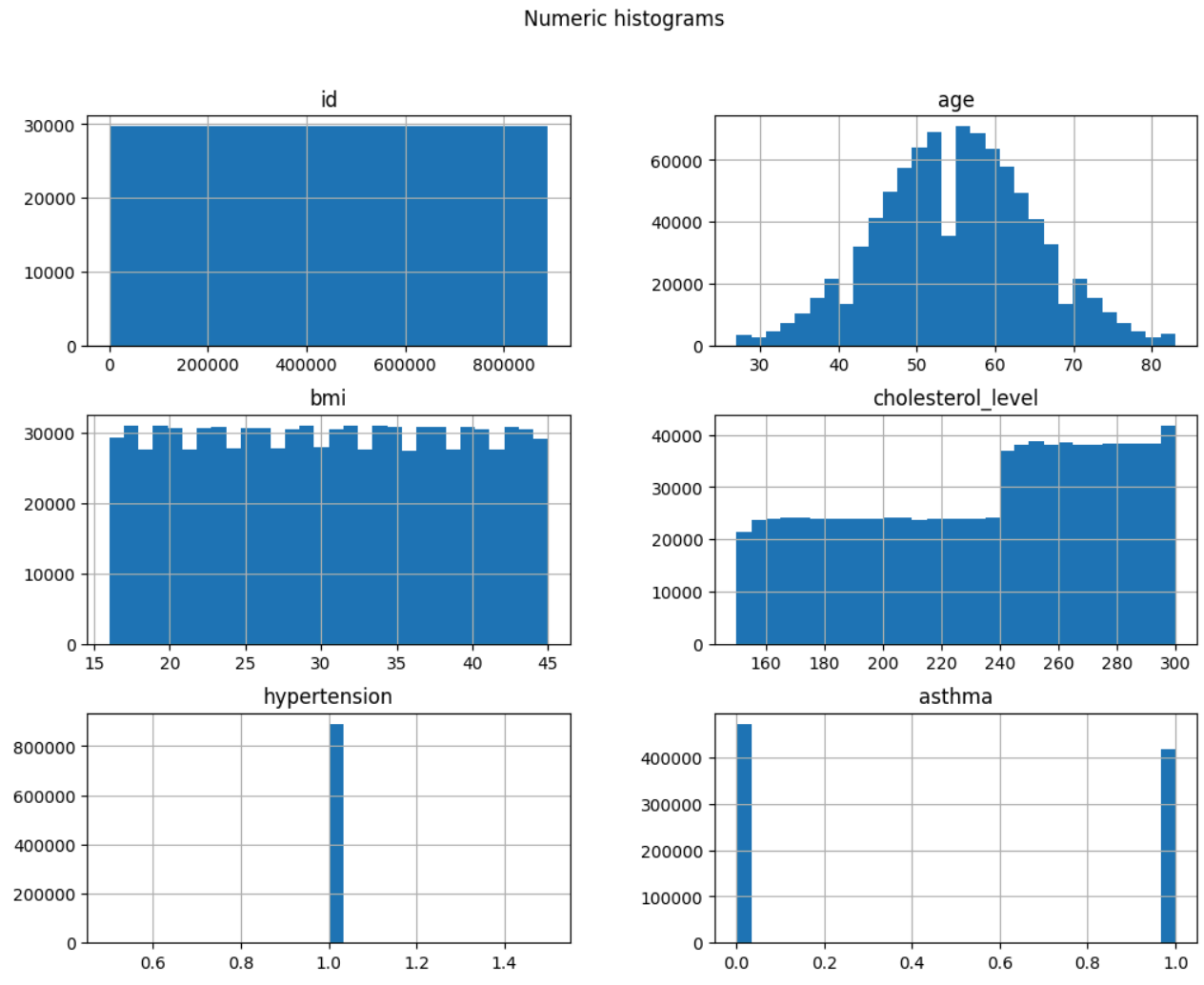
	id	age	gender	country	diagnosis_date	cancer_stage	family_history	smoking_status	
id	1.000000	0.000135	0.000079	0.000671	-0.001672	-0.001202	-0.000245	0.000236	-0.
age	0.000135	1.000000	-0.000954	-0.000316	0.000856	0.000912	-0.001044	-0.000013	0.
gender	0.000079	-0.000954	1.000000	0.001468	0.001933	-0.000161	0.000762	-0.000595	0.
country	0.000671	-0.000316	0.001468	1.000000	0.001281	-0.000665	-0.000624	0.001484	-0.
diagnosis_date	-0.001672	0.000856	0.001933	0.001281	1.000000	-0.000429	-0.000518	0.000490	-0.
cancer_stage	-0.001202	0.000912	-0.000161	-0.000665	-0.000429	1.000000	0.000345	0.001160	-0.
family_history	-0.000245	-0.001044	0.000762	-0.000624	-0.000518	0.000345	1.000000	-0.001261	0.
smoking_status	0.000236	-0.000013	-0.000595	0.001484	0.000490	0.001160	-0.001261	1.000000	0.
bmi	-0.001144	0.000955	0.000504	-0.000766	-0.000474	-0.000446	0.000411	0.000901	1.
cholesterol_level	-0.001507	0.001814	0.000224	-0.000483	-0.001084	-0.000504	0.000086	0.001169	0.
hypertension	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
asthma	0.000096	-0.000552	0.001142	-0.000521	-0.001362	-0.001350	-0.000808	-0.000140	0.
cirrhosis	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
other_cancer	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
treatment_type	0.000369	-0.001032	0.000040	0.001365	0.000915	-0.002192	0.000310	-0.000910	0.
end_treatment_date	0.000207	-0.000936	-0.000081	0.000043	0.001885	0.000992	0.001249	0.001188	-0.
survived	0.000517	0.001271	0.000762	0.000066	-0.002406	0.002519	0.001322	0.000087	0.

```
sns.heatmap(c,cmap='coolwarm')
```

```
<Axes: >
```




```
if len(num_cols)>0:
    cols = num_cols[:6]
    df1[cols].hist(bins=30, layout=(3,2), figsize=(12,9)); plt.suptitle('Numeric histograms'); plt.show()
```




```
for c in num_cols[:4]:
```

```
plt.figure(); sns.boxplot(x=df_encoded[c]); plt.title('Boxplot ' + c); plt.show()
```

```
X.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'],
      dtype='object')
```

```
y
```

	survived
0	0
1	1
2	0
3	0
4	0
...	...
889995	0
889996	0
889997	1
889998	0
889999	0

890000 rows × 1 columns

dtype: int64

```
df_encoded.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 890000 entries, 0 to 889999
Data columns (total 17 columns):
 #   Column              Non-Null Count   Dtype  
---  ------              --------------   -----  
 0   id                  890000 non-null  int64  
 1   age                 890000 non-null  float64
 2   gender              890000 non-null  int64  
 3   country             890000 non-null  int64  
 4   diagnosis_date      890000 non-null  int64  
 5   cancer_stage        890000 non-null  int64  
 6   family_history      890000 non-null  int64  
 7   smoking_status      890000 non-null  int64  
 8   bmi                 890000 non-null  float64
 9   cholesterol_level   890000 non-null  int64  
 10  hypertension        890000 non-null  float64
 11  asthma              890000 non-null  int64  
 12  cirrhosis           890000 non-null  float64
 13  other_cancer        890000 non-null  float64
 14  treatment_type      890000 non-null  int64  
 15  end_treatment_date  890000 non-null  int64  
 16  survived            890000 non-null  int64  
dtypes: float64(5), int64(12)
memory usage: 115.4 MB
```

```
from sklearn.feature_selection import SelectKBest,f_classif
selector = SelectKBest(score_func=f_classif, k=13)

X_new = selector.fit_transform(X, y)
selected_features = X.columns[selector.get_support()]

print("Top Selected Features:")
print(selected_features)
```

```
Top Selected Features:
Index(['id', 'age', 'gender', 'country', 'diagnosis_date', 'cancer_stage',
       'family_history', 'smoking_status', 'bmi', 'cholesterol_level',
       'asthma', 'treatment_type', 'end_treatment_date'],
      dtype='object')
```

```
print(y.value_counts())
```

```
survived
0    693996
1    196004
Name: count, dtype: int64
```

```
x_train,x_test,y_train, y_test =train_test_split(X,y,test_size=0.2,random_state=42)
```

```
ss=StandardScaler()
x_train_scaled=ss.fit_transform(x_train)
x_test_scaled=ss.transform(x_test)
```

```
models = {
    'LogisticRegression': LogisticRegression(),
    'RandomForest': RandomForestClassifier(),
    'GradientBoosting': GradientBoostingClassifier(random_state=42)
}
```

```
print(y_train.value_counts())
```

```
survived
0    555357
1    156643
Name: count, dtype: int64
```

```
results = {}
for name, model in models.items():
    print('\nTraining', name)
    model.fit(x_train_scaled, y_train)
    preds = model.predict(x_test_scaled)
    acc = accuracy_score(y_test, preds)
    print(f'Accuracy on test set: {acc:.4f}')
    print('Classification report:')
    print(classification_report(y_test, preds))
    results[name] = {'model': model, 'accuracy': acc}
res_df = pd.DataFrame({k: {'accuracy': v['accuracy']} for k,v in results.items()}).T.sort_values('accuracy', ascending=Fals
print('\nModel comparison:')
display(res_df)
```

```
Training LogisticRegression
Accuracy on test set: 0.7789
```