TITANIC3 - missing data

Required imports

```
import pandas as pd
import numpy as np
```

Read file

Missing values are marked as '?' in this file, so while reading the file I replaced them with NaNs to make dataset easier to work with. Since file is in arff format, I change names of columns manually.

```
data = pd.read_csv('/content/Zbiór danych Titanic.csv', header=None, na_values = "?")
```

```
columns = ['pclass', 'survived', 'name', 'sex', 'age', 'sibsp', 'parch', 'ticket', 'fare', 'cabin', 'embarked', 'boat', 'body', 'home.dest']
```

```
data.columns = columns
```

EDA

There are 14 features in dataset:

1. **Pclass** - Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd) - ordinal variable
2. **survival** - Survival (0 = No; 1 = Yes) - binary variable
3. **name** - Name - text variable (*from this variable we can get additional information such as marital status or family name*)
4. **sex** - Sex - categorical variable
5. **age** - Age - numerical variable (*age of children below 1 year is with precision of 1 month*)
6. **sibsp** - Number of Siblings/Spouses Aboard - numerical variable (dicrete)
7. **parch** - Number of Parents/Children Aboard - numerical variable (dicrete)
8. **ticket** - Ticket Number - text variable (*families have the same number of ticket apparently*)
9. **fare** - Passenger Fare (British pound) - numerical variable
10. **cabin** - Cabin - text variable (*cabins are assigned to a ticket, from some outer sources letter at the beginning means level with A being closest to deck and number is number of cabin on that level*)
11. **embarked** - Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton) - categorical variable
12. **boat** - Lifeboat - text variable
13. **body** - Body Identification Number - numerical variable (discrete)
14. **home.dest** - Home/Destination - text variable

```
data.head(n = 20)
```

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin
0	1	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5
1	1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26
2	1	0	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26
3	1	0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26
4	1	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	113781	151.5500	C22 C26
5	1	1	Anderson, Mr. Harry	male	48.0000	0	0	19952	26.5500	E12
6	1	1	Andrews, Miss. Kornelia Theodosia	female	63.0000	1	0	13502	77.9583	D7
7	1	0	Andrews, Mr. Thomas Jr	male	39.0000	0	0	112050	0.0000	A36
			Appleton, Mrs. Edward							

Next steps: View recommended plots

```
data.shape
```

```
(1309, 14)
```

```
data[data.duplicated()]
```

pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat	bo

Missing Values

There are 7 columns with missing values, 2 of them ('fare' and 'embarked') have less than 1% of values missing. For the other 5 ('age', 'cabin', 'boat', 'body', 'home.dest') NaNs account for more than 20% of all values. Features 'body' and 'cabin' stand out significantly here, for which the percentage is 90.76% and 77.46%, respectively. Some hipothesis to be checked in further analysis are:

1. 'embarked' and 'fare' missing values are completely random since they are individual cases - might be an oversight on the part of the crew
2. 'boat' variable identifies lifeboat which passenger boarded, possibly NaN values mean that passengers didn't board any - therefore did not survive
3. 'body' is Body Identification Number, those that survived and those whose bodies were not found were not assigned this number
4. 'cabin' might have more missing values for cheaper tickets and lower passenger class
5. 'age' and 'home.dest' to be checked for any correlations

```
summary = pd.DataFrame({
    'Data Type': data.dtypes,
    'Unique Values': data.nunique(),
    'NaN Count': data.isnull().sum(),
    'NaN Percentage': data.isnull().mean()
})

print(summary)
```

```
           Data Type  Unique Values  NaN Count  NaN Percentage
pclass         int64              3          0        0.000000
survived       int64              2          0        0.000000
name          object           1307          0        0.000000
sex           object              2          0        0.000000
age          float64             98        263        0.200917
sibsp          int64              7          0        0.000000
parch          int64              8          0        0.000000
ticket        object            929          0        0.000000
fare         float64            281          1        0.000764
cabin         object            186       1014        0.774637
embarked      object              3          2        0.001528
boat          object             27        823        0.628724
body         float64            121       1188        0.907563
home.dest     object            369        564        0.430863
```

'embarked' NaNs

There are only two records without information about port of embarkation. What is interesting, is that both passengers have the same ticket number. They could be family or friends. Both of them are women from first class. My guess is that while their boarding there must have been some oversight and this information wasn't taken down.

I would classify this as MCAR.

I would either drop those two rows since it should have much influence on quality of dataset and results of model or impute it with most common value for this variable which would be "S".

```
data[data['embarked'].isnull()]
```

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked
168	1	1	Icard, Miss. Amelie	female	38.0	0	0	113572	80.0	B28	NaN

```
data['embarked'].value_counts()
```

```
embarked
S    914
C    270
Q    123
Name: count, dtype: int64
```

'fare' NaNs

There is only one row with no information about fare that passenger paid for a ticket. It is no stowaway since there is number of his ticket. I would guess once again oversight of crew member.

I would classify this as MCAR.

Again, I would either drop this row or replace it with median value of fare for 3rd class passengers (not mean since there are some outliers).

```
data[data['fare'].isnull()]
```

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked
			Storey,								

```
data[data['pclass'] == 3]['fare'].describe()
```

```
count    708.000000
mean      13.302889
std       11.494358
min        0.000000
25%        7.750000
50%        8.050000
75%       15.245800
max       69.550000
Name: fare, dtype: float64
```

'body' NaNs

As we can see from belown result, everyone that survived has NaN as 'body' value, which confirms part of my hipothesis. Part about not found bodies can't be confirmed using data fromthis dataset, but doing some research I found information that only one in five bodies were recovered from ocean, which follows the statistics below.

I would classify this as MNAR.

Those rows definitely cannot be dropped since they are majority of records for passengers that didn't survived. Also imputing them with some numbers is not good solution since we are losing information. I would replace NaNs with label like 'unknown' to keep information about bodies not being found.

```
data.groupby('survived')['body'].apply(lambda x: x.isnull().mean())
```

```
survived
0    0.850433
1    1.000000
Name: body, dtype: float64
```

'boat' NaNs

Hypothesis about missing values for 'boat' is confirmed since almost everyone who had information about lifeboat survived and almost everyone who didn't have this information died. There are some exception but it is possible to die on a boat (freezing, falling out of the boat) or to survive in ocean (mostly great luck).

I would classify this as MNAR.

I would replace NaNs with label such as 'unknown' or 'missing' to distict those records.

```
data.groupby('survived')['boat'].apply(lambda x: x.isnull().mean())
```

```
survived
0    0.988875
1    0.046000
Name: boat, dtype: float64
```

'cabin' NaNs

Variable 'survived' is not that significant for missing values in 'cabin', although percentage of NaNs is higher for those who didn't survived. This is correlated to the fact that those who survived generally were from higher class and paid more for the tickets. There is almost no values in cabin variable for people from 2nd and 3rd class - probably they weren't that important for crew since they paid much less for the cruise.

I would classify this as MNAR.

I would replace NaNs with label such as 'unknown' or 'missing' to distict those records.

```
# percentage of nans in cabin column for groups from survived
data.groupby('survived')['cabin'].apply(lambda x: x.isnull().mean())
```

```
survived
0    0.873918
1    0.614000
Name: cabin, dtype: float64
```

```
# mean of survivors for each class
data.groupby('pclass')['survived'].mean()
```

```
pclass
1    0.619195
2    0.429603
3    0.255289
Name: survived, dtype: float64
```

```
# percentage of nans in cabin column for groups from pclass
data.groupby('pclass')['cabin'].apply(lambda x: x.isnull().mean())
```

```
pclass
1    0.207430
2    0.916968
```

```
3    0.977433
Name: cabin, dtype: float64
```

```
# median of fare paid for tickets for those who have information about cabin and who don't
data.groupby(data['cabin'].isnull())['fare'].median()
```

```
cabin
False    57.0
True     10.5
Name: fare, dtype: float64
```

'age' NaNs

There is small relationship between missing values for 'age' and surviving catstrophy or passenger class. However, if we look at records that don't have information about age, almost all of them don't have other variables such as 'cabin' or 'home.dest' which means we know less about those people. Also almost no body of person who died and whose age is unknown was found. Maybe age was collected after sinking of Titanic either from survivors or families of people whose bodies were recovered. This would also explain why there are no missing values for such informations as 'name' or 'sex' but 'age' which is similar those missing values has.

I would classify this as MAR.

Age is important variable if we want to predict who would survive Titanic therefore I would impute those NaNs. Children and older people had priority to enter lifeboats. To keep statistics accurate, I would group data by pclass and sex and then impute with median for each group.

```
data.groupby('survived')['age'].apply(lambda x: x.isnull().mean())
```

```
survived
0    0.234858
1    0.146000
Name: age, dtype: float64
```

```
data.groupby('pclass')['age'].apply(lambda x: x.isnull().mean())
```

```
pclass
1    0.120743
2    0.057762
3    0.293371
Name: age, dtype: float64
```

```
data[data['age'].isnull()].isnull().mean()
```

```
pclass       0.000000
survived     0.000000
name         0.000000
sex          0.000000
age          1.000000
sibsp        0.000000
parch        0.000000
ticket       0.000000
fare         0.000000
cabin        0.912548
embarked     0.000000
boat         0.737643
body         0.996198
home.dest    0.771863
dtype: float64
```

'home.dest' NaNs

There are more people without information about home/destination that did not survived, than those who did. Also most of them are from 3rd class. I think that this is a result of neglecting those people by crew and not gathering those information. Most people who died were from 3rd class therefore there is correlation between 'survived' variable and NaNs in 'home.dest'.

I would classify this as MAR.

I would replace those missing values with label 'unknown'.

```
data.groupby('survived')['home.dest'].apply(lambda x: x.isnull().mean())
```

```
survived
0    0.508035
```

```
1    0.306000
Name: home.dest, dtype: float64
```

```
data.groupby('pclass')['home.dest'].apply(lambda x: x.isnull().mean())
```

```
pclass
1    0.105263
2    0.057762
3    0.724965
Name: home.dest, dtype: float64
```

```
data[data['home.dest'].isnull()].isnull().mean()
```

```
pclass       0.000000
survived     0.000000
name         0.000000
sex          0.000000
age          0.359929
sibsp        0.000000
parch        0.000000
ticket       0.000000
fare         0.001773
cabin        0.934397
embarked     0.001773
boat         0.742908
body         0.914894
home.dest    1.000000
dtype: float64
```

```
data.groupby('pclass')['survived'].mean()
```

```
pclass
1    0.619195
2    0.429603
3    0.255289
Name: survived, dtype: float64
```

Kardynalność cech

```
print('Number of distinct labels for sex: {}'.format(len(data['sex'].unique())))
print('Number of distinct labels for ticket: {}'.format(len(data['ticket'].unique())))
print('Number of distinct labels for cabin: {}'.format(len(data['cabin'].unique())))
print('Number of distinct labels for embarked: {}'.format(len(data['embarked'].unique())))
print('Number of distinct labels for boat: {}'.format(len(data['boat'].unique())))
print('Number of distinct labels for home.dest: {}'.format(len(data["home.dest"].unique())))
```

```
Number of distinct labels for sex: 2
Number of distinct labels for ticket: 929
Number of distinct labels for cabin: 187
Number of distinct labels for embarked: 4
Number of distinct labels for boat: 28
Number of distinct labels for home.dest: 370
```

```
print('Number of passengers: {}'.format(len(data['name'].unique())))
```

```
Number of passengers: 1307
```

Low-cardinality

- sex - only two labels, binary variable
- embarked - after removing missing values there would be only three labels, which mean three cities passengers could onboard Titanic

High-cardinality

- ticket - many labels (929) which stand for number of ticket for given passenger, could be reduced for example to how many people were assigned to one ticket (were travelling together)
- cabin - many labels (187), could be reduced to level on which cabin is located, also maybe to part of ship like left, right, middle (would have to look for additional info like plan of all cabins to find a pattern)
- home.dest - many labels (370), could be reduced to country instead of city, I think it would reduce number of labels significantly

Standard-cardinality (?)

- boat - has 28 labels which is much less than for example ticket or home.dest but it is not as few as 3 for embarked or 2 for sex, I don't see a way to reduce it

Reduce cardinality for 'cabin' variable

```
print(f"Type of returned value: {type(data['cabin'].unique())}")
print(f"Number of unique labels for 'cabin' variable: {len(data['cabin'].unique())}")
```

```
Type of returned value: <class 'numpy.ndarray'>
Number of unique labels for 'cabin' variable: 187
```

```
data["CabinReduced"] = data['cabin'].map(lambda x: x[0] if isinstance(x, str) else np.nan)
display(data[["cabin", "CabinReduced"]].head(20))
```

	cabin	CabinReduced
0	B5	B
1	C22 C26	C
2	C22 C26	C
3	C22 C26	C
4	C22 C26	C
5	E12	E
6	D7	D
7	A36	A
8	C101	C
9	NaN	NaN
10	C62 C64	C
11	C62 C64	C
12	B35	B
13	NaN	NaN
14	A23	A
15	NaN	NaN
16	B58 B60	B
17	B58 B60	B
18	D15	D
19	C6	C

```
print('Number of labels for CabinReduced: {}'.format(len(data['CabinReduced'].unique())))
print(f'Cardinality lowered by {(len(data.cabin.unique()) - len(data.CabinReduced.unique())) / len(data.cabin.unique()) * 100} %')
```

```
Number of labels for CabinReduced: 9
Cardinality lowered by 95.18716577540107 %
```

I am reducing cardinality of this variable because it is too granular for model to work properly. Such high number of labels make model way too complex and reduces explainability. It could also lead to overfitting my model and lead to low performance on test dataset since the model will have trouble adapting to new data. However, if I want to reduce cardinality this way, I am losing potentially important information about side of Titanic the cabin was located on. Information about who lived with who also is lost but I don't think it is significant in case of chances of surviving. I would suggest way of coding data where first part would level of ship the cabin was on and then left, right, middle. This way number of labels still wouldn't be high - 27, but I would keep this potentially important information in my dataset.