Missing Values

Gözlemlerde eksiklik olmasını durumunu ifade etmektedir.

Araç Fiyatı	KM	Vites Türü	Hasar Durumu	Marka	Model
10000	300000	Manuel	Evet	A	A1
54000	40000	Manuel	Hayır	A	A2
46999	NA	Manuel	Evet	B	B1
89000	1000000	Otomatik	Evet	C	C1
70000	78000	Otomatik	Evet	B	B2
50000	30000	Manuel	Hayır	C	C2
NA	600000	Manuel	Hayır	C	C3
68900	50000	Otomatik	Hayır	B	B2
12000	200000	Manuel	Hayır	A	A1

Eksik veri problemi nasıl çözülür ?

- Silme
- Değer Atama Yöntemleri (Basit Atama Yöntemleri)

  Gelişmiş veya tahminsel değil, daha basit bir şekilde ortalama ve medyan gibi değerler ile atama işlemlerini gerçekleştirir.
- Tahmine Dayalı Yöntemler

  Makine öğrenmesi veya istatistiksel bazı tahminlere göre değer atamaya dayanmaktadır.

Eksik veriler ile çalışırken göz önünde bulundurulması gereken önemli konulardan birisi :

Eksik verinin rassallığı, yani eksikliğin rastgele çıkıp çıkmadığı durumunun bilinmesi gerekmesidir.

Eksik değerlere sahip gözlemlerin veri setinden direkt çıkarılması ve rassallığının incelenmemesi, yapılacak istatistiksel çıkarımların ve modelleme çalışmalarının güvenilirliğini düşürür.(Alpar, 2011)

Eksik gözlemlerin veri setinden direkt çıkarılabilmesi için veri setindeki eksikliğin bazı durumlarda kısmen bazı durumlarda tamamen rastlantısal olarak oluşmuş olması gerekmektedir. Eğer eksiklikler değişkenler ile ilişkili olarak ortaya çıkan yapısal problemler ile meydana gelmişse bu durumda yapılacak silme işlemi ciddi yanlılıklara sebep olabilecektir.(Tabachnick & Fidell, 1996)

Kredi kartı harcamalarını ifade eden bir değişkenin incelendiği ve bu değişkenin bazı değerlerinin Na olduğu varsayılsın.

Eğer bu NA değerleri rastgele ortaya çıktıysa silme işlemi, basit atama yöntemi veya tahmine dayalı yöntem rahat bir şekilde uygulanabilir.

Eksik değerlerin bağımlı olması, rastgele olmaması ne demektir ?

Kredi kartının var olup olmamasını ifade eden bir değişken ile kredi kartı harcamalarını ifade eden değişkenin incelendiği varsayılsın.

Bir kişinin kredi kartı yoksa, kredi kartı harcaması da olamayacaktır, yani değeri 0 olacaktır.

Ve 0 nümerik olarak basılmayacağı için NA olarak kaydedilecektir.

Dolayısıyla bu durum rastgele değil, kredi kartı olup olmama durumunu ifade eden başka bir değişkene bağlıdır.

Bu duruma “Bir değişkendeki eksiklik başka bir değişken etkisinde ortaya çıkmıştır.” yorumu yapılır. Bu sebeple kredi kartı harcaması değişkenindeki eksiklikler rasgele değildir.

Eksik Değerleri Yakalama

```
# Veri setinin tamamında eksik veri olup olmadığını kontrol etmek için :
df.isnull().values.any()

# Değişkenlerdeki eksik değer sayısına erişmek için :
df.isnull().sum()

# Değişkenlerdeki tam(eksik olmayan) değer sayısına erişmek için :
df.notnull().sum()

# Veri setindeki toplam eksik değer sayısına erişmek için :
df.isnull().sum().sum()

# En az bir tane eksik değere sahip olan gözlem birimlerine erişmek için :
df[df.isnull().any(axis=1)]

# Tam olan gözlem birimlerine erişmek için :
df[df.notnull().any(axis=1)]

# Azalan şekilde sıralamak için :
df.isnull().sum().sort_values(ascending=False)

# Eksik değerlerin tüm veri setine oranı :
(df.isnull().sum() / df.shape[0] * 100).sort_values(ascending=False)  # yüzdelik olarak eksiklik oranlarına erişilir.

# Yalnızca eksik değere sahip olan değişkenleri gözlemlemek için :
na_cols = [col for col in df.columns if df[col].isnull().sum() > 0]
```

Bu işlemlerin fonksiyonlaştırılması :

```
def missing_values_table(dataframe, na_name=False):
    na_columns = [col for col in df.columns if df[col].isnull().sum() > 0]
    n_miss = dataframe[na_columns].isnull().sum().sort_values(ascending=False)
    ratio = (dataframe[na_columns].isnull().sum() / dataframe.shape[0] * 100).sort_values(ascending=False)
    missing_df = pd.concat([n_miss, np.round(ratio, 2)], axis=1, keys=['n_miss', 'ratio'])  # Oluşturulan bu iki değişken sütunlara göre birleştirilir.
    print(missing_df, end="\n")

    if na_name:  # Eksik değerlere sahip değişkenlerin isimlerine erişmek için na_name=True.
        return na_columns
```

```
missing_values_table(df)
missing_values_table(df, True)
```

Bu fonksiyon ile :

- Bir verideki eksikliklerin frekans bilgisine
- Eksikliklerin hangi değişkenlerde olduğu bilgisine
- Bu eksikliklerin oranı bilgisine erişilebilir.

Eksik Değer Problemini Çözme

Bir değişkenin boş gözlemleri

- O değişkenin ortalaması ile
- Medyanı ile
- Herhangi sabit bir değer ile doldurulabilir.

Çözüm 1 : Silme İşlemi

```
df.dropna().shape
```

Çözüm 2 : Basit Atama Yöntemleri ile Doldurma İşlemi

```
# Yaş değişkeninin boş değerlerini, ortalaması ile doldurmak için :
df["Age"].fillna(df["Age"].mean()).isnull().sum()

# Yaş değişkeninin boş değerlerini, medyanı ile doldurmak için :
df["Age"].fillna(df["Age"].median()).isnull().sum()

# Yaş değişkeninin boş değerlerini herhangi sabit bir değer ile doldurmak için (mesela 0)
df["Age"].fillna(0).isnull().sum()
```

Bu işlemin lambda ile programatik bir şekilde yapılaması :

```
dff = df.apply(lambda x: x.fillna(x.mean()) if x.dtype != "O" else x, axis=0).head()
dff.isnull().sum().sort_values(ascending=False)  # Burada kategorik değişkenlerde hala Na değerleri olduğu gözlemlenir.
```

Kategorik değişkenlerde Na değerlerinden kurtulmanın en mantıklı yolu , Na değerlerini o değişkenin modu ile dordurmaktır.

```
df["Embarked"].fillna(df["Embarked"].mode()[0]).isnull().sum()  # Na değerini Değişkenin modu ile doldurma işlemi.
df["Embarked"].fillna("missing")  # Na değerini herhangi sabit bir değer ile doldurma.
```

Kategorik değişkenlerdeki Na değerleri de lambda ile programatik bir şekilde doldurulabilir :

```
df.apply(lambda x : x.fillna(x.mode()[0]) if (x.dtype == "O" and len(x.unique()) <= 10) else x, axis=0).isnull().sum()
```

Kategorik Değişken Kırılımında Değer Atama

Yukarıdaki işlemlerde sayısal ve kategorik değişkenlerinin sahip olduğu Na değerlerinin yerine bu değişkenlerin mean, mode gibi değerleri atandı.

Buradaki amaç :

Veri setinde var olan bazı kategorik değişkenleri kırılım olarak ele almak ve bu kırılımlar neticesinde elde edilecek değerleri ilgili boşluklara atamak.

```
df["Age"].mean(). # Yaş değişkeninin ortalaması
df.groupby("Sex")["Age"].mean()  # Cinsiyet kırılımında Yaş değişkeninin ortalaması.
```

Daha önce "Age" değişkeninin Na değerleri direkt "Age" değişkeninin ortalaması(29.69) ile doldurulmuştu.

Burada ise :

Tüm eksikliklere 29.69 atamak yerine cinsiyete göre bir kırılıp oluşturup,

female Na değerleri için female yaş ortalaması atanması,

male Na değerleri için male yaş ortalaması atanması daha doğru olacaktır.

Bu işlemlerin programatik bir şekilde yapılması :

```
df["Age"].fillna(df.groupby("Sex")["Age"].transform("mean")).isnull().sum()
```

.transform : Kırılımla gelen cinsiyete göre ortalama değerlerini ilgili boşluklara uygular.

Bu işlemin farklı bir gösterimi :

```
# Yaş değişkeninde Na olup cinsiyeti female olanları yaş değişkeninin kendisi ile getirir :
df.loc[(df["Age"].isnull()) & (df["Sex"] == "female"), "Age"]
```

```
# groupby'a göre ortalama işleminin female'ler için olan kısmına erişilmek istenirse :
df.groupby("Sex")["Age"].mean()["female"]

# female'deki Na değerleri, female'in mean değerleri ile dolduruldu.
df.loc[(df["Age"].isnull()) & (df["Sex"] == "female"), "Age"] = df.groupby("Sex")["Age"].mean()["female"]

# Aynı işlem male için de yapılabilir.
df.loc[(df["Age"].isnull()) & (df["Sex"] == "male"), "Age"] = df.groupby("Sex")["Age"].mean()["male"]

df.isnull().sum()
```

Çözüm 3: Tahmine Dayalı Atama ile Doldurma İşlemi

Bu çözüm yönteminde, Makine Öğrenmesi yöntemi ile tahmine dayalı bir şekilde modelleme işlemi gerçekleştirilecektir.

Eksikliğe sahip olan değişken bağımlı değişken, diğer değişkenler bağımsız değişkenler olarak kabul edilip modelleme işlemi gerçekleştirilecektir ve bu modelleme işlemine göre Na olan değerler tahmin edilmeye çalışılacaktır.

Bu konudaki kritik noktalar :

1. Modelleme tekniği kullanılacak olması dolayısıyla bu modelin beklediği standartların karşılanması gerekmektedir. Bu sebeple kategorik değişkenlerin One-Hot Encoder işleminden geçmesi gerekir.

   ```
   cat_cols, num_cols, cat_but_car = grab_col_names(df)

   num_cols = [col for col in num_cols if col not in "PassengerId"]
   ```

   Buradaki cat_cols'lara bir dönüşüm işlemi (Label Encoding ya da One Hot Encoder) yapılması gerekmektedir.

   Label Encoding ve One Hot Encoder işleminin aynı anda yapılabilmesi için One Hot Encoder'ın uygulanacağı get_dummies methodu kulanılır :

   ```
   dff = pd.get_dummies(df[cat_cols + num_cols], drop_first=True)
   ```

   drop_first argümanının True olarak kullanılması, 2 sınıfa sahip olan değişkenlerin ilk sınıfını atar ve 2. sınıfını tutmaya yarar.

   Burada özetle yapılan işlem 2 ya da daha fazla sınıfa sahip kategorik değişkenleri nümerik bir şekilde ifade etmektir.

   get_dummies methodu tüm değişkenler birlikte verilse dahi sadece kategorik değişkenlere dönüşüm uygulamaktadır. Bu sebeple kullanılacak olan değişkenler cat_cols + num_cols şeklinde bir araya getirilebilirler.

   Bu işlem sonucu veri, kullanıcalacak olan makine öğrenmesi yönteminin anlayacağı şekilde dönüştürülmüştür.

2. KNN uzaklık temelli bir algoritma olduğundan dolayı değişkenlerin standartlaştırılması gerekir :

   ```
   scaler = MinMaxScaler(). # MinMaxScaler, değerleri 0 ile 1 arasına dönüştürme bilgisini taşır.
   dff = pd.DataFrame(scaler.fit_transform(dff), columns=dff.columns)
   # MinMaxScaler dönüştürme işlemi, scaler.fit_transform(dff) ile veri setine uygulanır.
   # MinMaxScaler sonucu gelen veri istenilen formatta olmadığından dataframe'e çevrilir.
   # Değişken isimleri dff.columns=dff.columns ile alınır.
   ```

   Böylece Makine Öğrenmesi tekniği kullanılabilmesi için veri uygun hale getirildi.

   KNNImputer Nasıl Çalışır ?

   Na değere sahip değişkenlerde :

   Bu değişkendeki Na değerini doldurmaki için Na olmayan gözlem birimlerinde bulunan en yakın komşularının ortalaması alınır ve bu ortalama değeri Na değere atanır.

   Komşu sayısını n_neighbors parametresi belirler.

   KNN'in uygulanması :

   ```
   from sklearn.impute import KNNImputer  # Makine öğrenmesi teknigi ile tahmine dayalı bir şekilde eksik değerleri doldurma imkanı sağlayacaktır.
   imputer = KNNImputer(n_neighbors=5)  # Model nesnesi oluşturulur ve komşuluk sayısı 5 yapılır.
   dff = pd.DataFrame(imputer.fit_transform(dff), columns=dff.columns)  # KNNImputer, imputer.fit_transform ile veri setine uygulanır.
   dff.head()
   ```

   Na değerleri dolduruldu fakat kıyaslama ve inceleme işlemleri, atama işlemlerinin takibi nasıl gerçekleştirilir ?

   Kıyaslama yapma amaçlı standartlaştırma işlemini geri almak için :

   ```
   dff = pd.DataFrame(scaler.inverse_transform(dff), columns=dff.columns)
   dff.head()
   ```

   Bu kıyas işlemini yapabilmek için :

   ```
   df["age_imputed_knn"] = dff["Age"] # Na değerleri doldurulan değişken ilk df'e değişken olarak eklenir.
   df.loc[df["Age"].isnull(), ["Age", "age_imputed_knn"]]  # Age değişkeninin eksik değerleri ile Age ve age_imputed_knn gözlemlenir.
   df.loc[df["Age"].isnull()]  # tüm değişkenleri incelemek için.
   ```

   Makine Öğrenmesi tekniği kullanılarak veri setindeki eksik olan değerlerin yerine tahmin edilen değerler atanmış olundu.

Yapılan işlemlerin özeti :

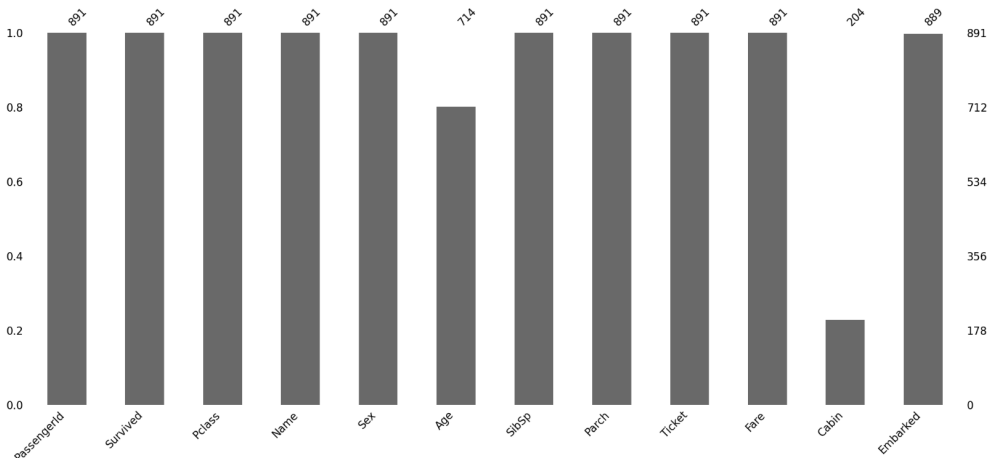
```
df = load()
# missing table
missing_values_table(df)
# sayısal değişkenleri direk median ile oldurma
df.apply(lambda x: x.fillna(x.median()) if x.dtype != "O" else x, axis=0).isnull().sum()
# kategorik değişkenleri mode ile doldurma
df.apply(lambda x: x.fillna(x.mode()[0]) if (x.dtype == "O" and len(x.unique()) <= 10) else x, axis=0).isnull().sum()
# kategorik değişken kırılımında sayısal değişkenleri doldurmak
df["Age"].fillna(df.groupby("Sex")["Age"].transform("mean")).isnull().sum()
# Tahmine Dayalı Atama ile Doldurma
```

Gelişmiş Analizler

Eksik Verinin Yapısını İncelemek

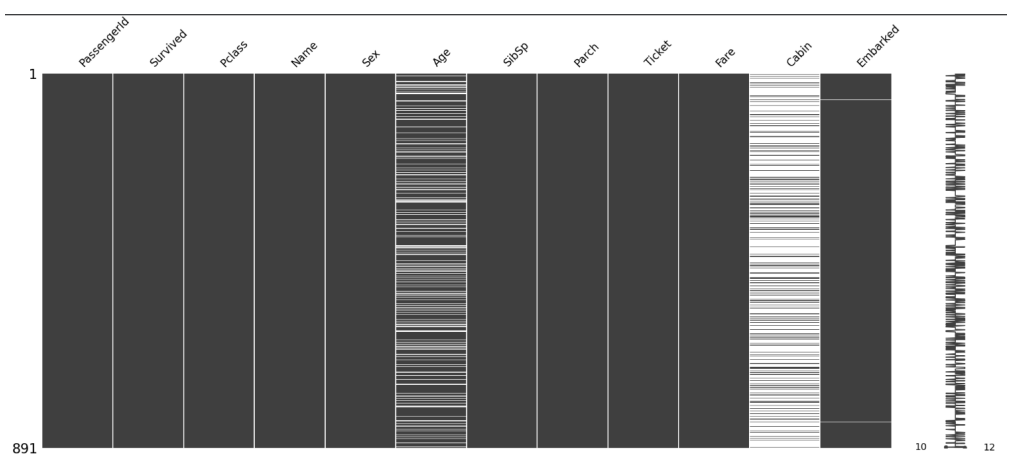
```
df = load()

msno.bar(df) # Eksik verinin yapısını incelemek için msno kütüphanesi df'e uygulanır.
plt.show()
```



Bar methodu burada, ilgili veri setindeki değişkenlerin tam olan gözlemlerin sayılarını verir.

```
msno.matrix(df). # Eksik değerleri gözlemleme.
plt.show()
```



msno.matrix :

Değişkenlerdeki eksik değerleri gözlemleme imkanı sağlayan görsel bir araçtır. Beyaz çizgiler, her sütundaki eksik değerleri gösterir.

Grafik daha yakından incelendiğinde, boşluklar arasında bir ilişki olup olmadığı değerlendirilebilir.

Bu ilişkiler, değişkenlerin birbiriyle nasıl ilişkili olduğu, birbirleri arasında bir bağımlılık söz konusu olup olmadığı hakkında bir fikir verir.

msno.heatmap :

Eksiklikler üzerine kurulu bir ısı haritasıdır. Bir değişkendeki eksik değerin diğer değişkenlerdeki eksik değerler ile nasıl bir ilişki içerisinde olduğunu anlamamızı sağlar.

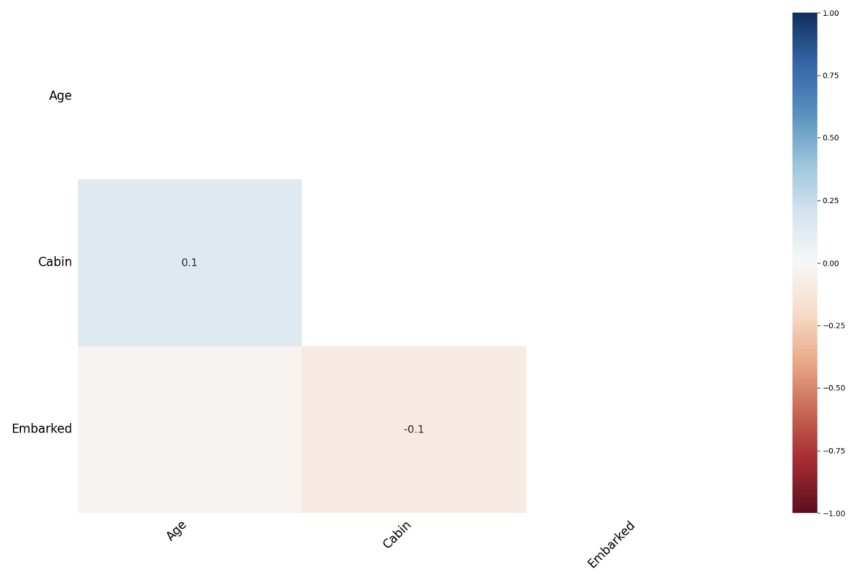
Aşağıdaki grafiğin sağ tarafında gözlemlenen palet nullity correlation'u temsil eder ve -1 ile 1 arasında değişir.

+1'e yakın değerler pozitif yönlü kuvvetli ilişkiyi, -1'e yakın değerler negatif yönlü kuvvetli ilişkiyi ifade eder.

Daha detaya inilecek olunursa :

- -1 : Tam Negatif korelasyon, bir değişkenin değeri varsa diğer değişkenin değerinin kesinlikle olmadığını gösterir.
- 0 : İlişki yok, değişkenlerin var olan veya olmayan değerlerinin birbirleri üzerinde herhangi bir etkisinin olmadığını gösterir.
- 1 : Tam Pozitif korelasyon, bir değişkenin değeri mevcutsa diğerinin değerinin de kesinlikle mevcut olduğunu ifade eder.

```
msno.heatmap(df)
plt.show()
```



Yukarıdaki grafiğe göre :

“Cabin” değişkeni ile “Age” değişkeni arasındaki eksiklik korelasyonu incelenecek olunursa bu değer 0.1’dir. Anlamsız bir korelasyondur.

Benzer şekilde “Embarked” ile “Cabin” değişkenleri arasındaki korelasyon incelenecek olunursa bu değer -0.1’dir. Bu da anlamsız bir korelasyondur.

Dolayısıyla, “bu veri setinde eksiklik korelasyonu anlamlı görünmüyor.” yorumu yapılabilir.

Eksik Değerlerin Bağımlı Değişken ile Analizi

Öncelikle eksik değerlere sahip değişkenler ve index bilgilerine daha önce oluşturulan `missing_values_table` fonksiyonu ile erişilir.

```
def missing_values_table(dataframe, na_name=False):
    na_columns = [col for col in df.columns if df[col].isnull().sum() > 0]
    n_miss = dataframe[na_columns].isnull().sum().sort_values(ascending=False)
    ratio = (dataframe[na_columns].isnull().sum() / dataframe.shape[0] * 100).sort_values(ascending=False)
    missing_df = pd.concat([n_miss, np.round(ratio, 2)], axis=1, keys=['n_miss', 'ratio'])  # Oluşturulan bu iki değişken sütunlara göre birleştirilir.
    print(missing_df, end="\n")

    if na_name:  # Eksik değerlerin barındığı değişkenlerin isimlerine erişmek için bu paratmetre True yapılır.
        return na_columns


missing_values_table(df, True)
na_cols = missing_values_table(df, True)
```

```
def missing_vs_target(dataframe, target, na_columns):  # target parametresi ilgili veri setindeki bağımlı değişkeni temsil eder.
    temp_df = dataframe.copy(). # dataframe'in bir kopyası oluşturulmuş.(opsiyonel)

    for col in na_columns:  # Na değerleri barındıran columnlar'da gezilir.
        temp_df[col + '_NA_FLAG'] = np.where(temp_df[col].isnull(), 1, 0)
    # 1- temp_df[col + '_NA_FLAG'] ile Na değere sahip değişkenlerin sonuna "_NA_FLAG" eklenerek yeni değişken oluşturuldu.
    # ve yukarıda oluşturulan yeni dataframe "temp_df" içerisine eklendi.
    # 2- np.where(temp_df[col].isnull(), 1, 0) :
    # ilgili değişken temp_df[col] ile seçildi.
    # Seçilmiş olan değişkende eksiklik olup olmadığı .isnull() ile kontrol edilir.
    # Na görülen değerleri temsil etmesi için 1, Na görülmeyen değerleri temsil etmesi için 0 atanır.


    na_flags = temp_df.loc[:, temp_df.columns.str.contains("_NA_")].columns. # Burada list comprehensions yöntemi de uygulanabilir.
    # temp_df.loc ile temp_df içerisinde bir seçim işlemi yapılır.
    # : ile tüm satırlar istenir.
    # Fakat temp_df.columns.str.contains("_NA_") ile sütünlar içerisinden "_NA_" ifadesini içerenler istenir.
    for col in na_flags:
        print(pd.DataFrame({"TARGET_MEAN": temp_df.groupby(col)[target].mean(),  # yeni değişkenlere göre gruplandırma işlemi yapılır ve target değişkenin ortalaması alınır.
                            "Count": temp_df.groupby(col)[target].count()}), end="\n\n\n")  # aynı zamanda target'in count değeri hesaplanıp gösterilir.

# "Survived" değişkeni ile eksikliğe sahip değişkenler karşılaştırılır :
missing_vs_target(df, "Survived", na_cols)
```

Eksik değer gözlemlenen 3 değişkenin, bağımlı değişken olan "Survived" değişkeninin ortalaması açısından değerlendirmeleri :

```
In [9]: missing_vs_target(df, "Survived", na_cols)
             TARGET_MEAN  Count
Age_NA_FLAG
0                  0.406    714
1                  0.294    177


              TARGET_MEAN  Count
Cabin_NA_FLAG
0                   0.667    204
1                   0.300    687


                 TARGET_MEAN  Count
Embarked_NA_FLAG
0                      0.382    889
1                      1.000      2
```

Eksik değere sahip noktalar 1 ile, dolu gözlemler ise 0 ile temsil edilmektedir.

Ana odak, "Survived" değişkenine göre, hayatta kalma durumunu ortaya çıkaran etkiyi yakalamaya çalışmaktır.

Yani bir modelleme aşamasına geçildiğinde, “Yolcuların bu gemiye bindiğinde hayatta kalma ya da kalamama oranlarını ortaya çıkaran etki nedir ?” sorusuna yanıt vermeye çalışılacak ve bazı yolcuların bilgileri verildiğinde hayatta kalıp kalamayacakları tahmin edilmeye çalışılacaktır.

1. **“Age_NA_FLAG” :**

   “Age_NA_FLAG” değişkeninde eksikliğin gözlendiği durumlarda hayatta kalma oranı 0.29’dur.

   “Age_NA_FLAG” değişkeninde eksikliğin gözlenmediği durumlarda hayatta kalma oranı 0.40’tır.

   177 eksik gözlem ve 714 eksik olmayan gözlem vardır. Count kısmındaki sayı dikkate değer ise bu değişkenin ortalaması ile yorumlar yapılır.

   Dolayısıyla çıktıya göre, ""“Age_NA_FLAG”" değişkenindeki eksikliğin gözlendiği durumda hayatta kalma oranı, eksiklik gözlenmeyen duruma göre daha düşüktür.

   Ama buradan bir sonuca gitmek zordur ve diğer değişkenler de incelenmelidir.

2. **“Cabin_NA_FLAG” :**

3. 687 eksik gözlem ve 204 eksik olmayan gözlem vardır.

   Değişkenin değerlerinin %75’inden fazlasının eksik olduğu gözlemlenir.

   Bu değişkenin atılması tercih edilebilirdi. **Fakat** değişkendeki Na değerlerin (Kabin numarası belli olmayan yolcular) bir bilgi verme ihtimaline karşı bu değişken incelemeye alınır.

   “Cabin_NA_FLAG” değişkeninde eksikliğin gözlendiği durumlarda "Survived" değişkeninin ortalamasına göre hayatta kalma oranı 0.30’dur.

   “Cabin_NA_FLAG” değişkeninde eksikliğin gözlenmediği durumlarda "Survived" değişkeninin ortalamasına göre hayatta kalma oranı 0.66’dır.

   Buraya ciddi bir şekilde odaklanmak gerektiği çok açıktır.

   Cabin numarası olmayanların hayatta kalma oranlarının daha düşük olduğu gözlemlenir.

   “Cabin_NA_FLAG” bilgisi Na olanlar bize bu senaryoda bir bilgi verdiğinden “Cabin_NA_FLAG” değişkenini silmek yerine feature extraction konusunda değişkene çevrilebilir.

3. **“Embarked_NA_FLAG” :**

   “Embarked_NA_FLAG” değişkeninde Na sayısı 2 dir. Yorumlanabilecek bir durum söz konusu değildir.

Özet

```
df = load()
na_cols = missing_values_table(df, True)
# sayısal değişkenleri median ile doldurma
df.apply(lambda x: x.fillna(x.median()) if x.dtype != "O" else x, axis=0).isnull().sum()
# kategorik değişkenleri mode ile doldurma
df.apply(lambda x: x.fillna(x.mode()[0]) if (x.dtype == "O" and len(x.unique()) <= 10) else x, axis=0).isnull().sum()
# kategorik değişken kırılımında sayısal değişkenleri doldurmak
df["Age"].fillna(df.groupby("Sex")["Age"].transform("mean")).isnull().sum()
# Tahmine Dayalı Atama ile Doldurma
missing_vs_target(df, "Survived", na_cols)
```

References:

• https://courses.miuul.com/p/feature-engineering?gclid=Cj0KCQiA1ZGcBhCoARIsAGQ0kkrOv2WlbA52S5pmw4iGYG5vLYUsMOEdaCWe79tQQlKqfHq5vk1IhV0aAjT2EALw_wcB

•https://coderzcolumn.com/tutorials/data-science/missingno-visualize-missing-data-in-python

https://github.com/ResidentMario/missingno