Python for Data Science and Machine Learning

School Year 2023-2024

IST

Course Structure

#	Topic	Dates
1	Introduction to Python	OCT 19, OCT 26, NOV 9, NOV 16, NOV 23, NOV 30, DEC 7
2	Python Challenge	DEC 14
3	Introduction to Data Science	DEC 21, JAN 11, JAN 18, JAN 25
4	Introduction to Machine Learning	FEB 1, FEB 22, FEB 29
5	Data Science & ML Challenge	MAR 7

= Core Topics = Optional Topics

Jupyter Notebook Setup

In a browser:

```
192.168.10.4:8888
```

Password: **ist**

Recap: Pandas & other LIbraries

Pandas is a powerful Python data analysis toolkit.

Matplotlib & **Seaborn** are plotting libraries.

13.0

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

I have added two functions (**plot_2d** & **plot_3d**) that will help plotting charts in future exercises

Recap: DataFrame

A **DataFrame** is a two-dimensional data structure with labeled axes (rows and columns).

13.1

```
df = pd.read_csv("titanic_dataset.csv")
df
```

Recap: DataFrame

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

Test & Training Data

The titanic **dataset** is split in two:

1. **Train data** (to build our Data Science/Machine Learning Models)
2. **Test data** (to evaluate our DS/ML Models)

13.2

```
test_df = pd.read_csv("test_dataset.csv")

(len(df), len(test_df))
```

Recap: Indexing, Grouping & Analysis

When using them all together, in order we:

1. First use boolean indexing
2. Secondly use grouping
3. Finally we select the analysis function we'd like

```
df[df["Age"] < 18].groupby("Pclass")["Survived"].count()
```

Indexing **Grouping** **Data Analysis**

Exploratory Data Analysis (EDA)

<u>Before</u> we dive into Machine Learning: EDA!

Exploratory Data Analysis refers to the critical process of performing initial **investigations on data** so as to discover **patterns**,to spot **anomalies**,to test hypothesis and to check **assumptions**.

Pratil, Prasad. (2018). "What is Exploratory Data Analysis?" Towards Data Science.
Available at: https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15.

Feature Engineering

Feature engineering or feature extraction or feature discovery is the process of **extracting features** (characteristics, properties, attributes) **from raw** data **to support training** a downstream statistical model.

Hastie, Trevor; Tibshirani, Robert; Friedman, Jerome H. (2009).
The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer. ISBN 978-0-387-84884-6.

Analysing the “Embarked” Column

We can see that not all passengers have data regarding their embarkation point:

13.3.1

```
df[pd.isna(df["Embarked"])]
```

```
df[pd.isna(df["Embarked"])]
```

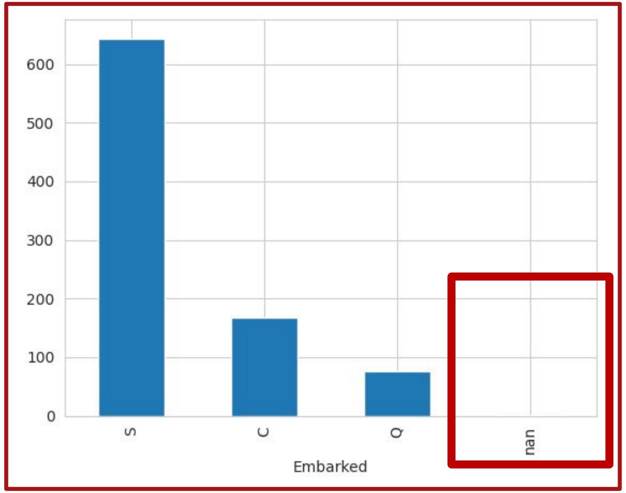
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
61	62	1	1	Icard, Miss. Amelie	female	38.0	0	0	113572	80.0	B28	NaN
829	830	1	1	Stone, Mrs. George Nelson (Martha Evelyn)	female	62.0	0	0	113572	80.0	B28	NaN

Analysing the “Embarked” Column

To visualise the current value distribution:

13.3.2

```
df['Embarked'].value_counts(dropna=False).plot(kind='bar')
```



Analysing the “Embarked” Column

Exercise 13.3.3: What value do we pick as default for rows missing data:

13.3.3

```
df['Embarked'] = df['Embarked'].fillna(value=________
```

Verify that the value was set correctly by running cell **13.3.4**

Solution 13.3.3

```
df['Embarked'] = df['Embarked'].fillna(value='S')
```

Analysing the “Age” Column

We can see that not all passengers have data on their age:

13.4.1

```
df[pd.isna(df["Age"])]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
17	18	1	2	Williams, Mr. Charles Eugene	male	NaN	0	0	244373	13.0000	NaN	S
19	20	1	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.2250	NaN	C
26	27	0	3	Emir, Mr. Farred Chehab	male	NaN	0	0	2631	7.2250	NaN	C
28	29	1	3	O'Dwyer, Miss. Ellen "Nellie"	female	NaN	0	0	330959	7.8792	NaN	Q
...	...	...	...	...	...	...	...	...	...	...	...	...
859	860	0	3	Razi, Mr. Raihed	male	NaN	0	0	2629	7.2292	NaN	C
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	NaN	8	2	CA. 2343	69.5500	NaN	S
868	869	0	3	van Melkebeke, Mr. Philemon	male	NaN	0	0	345777	9.5000	NaN	S
878	879	0	3	Laleff, Mr. Kristo	male	NaN	0	0	349217	7.8958	NaN	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S

177 rows × 12 columns

Analysing the "Age" Column

Exercise 13.4.2: Let's use a scatter plot to graphically determine which columns are correlated with **"Age"**.

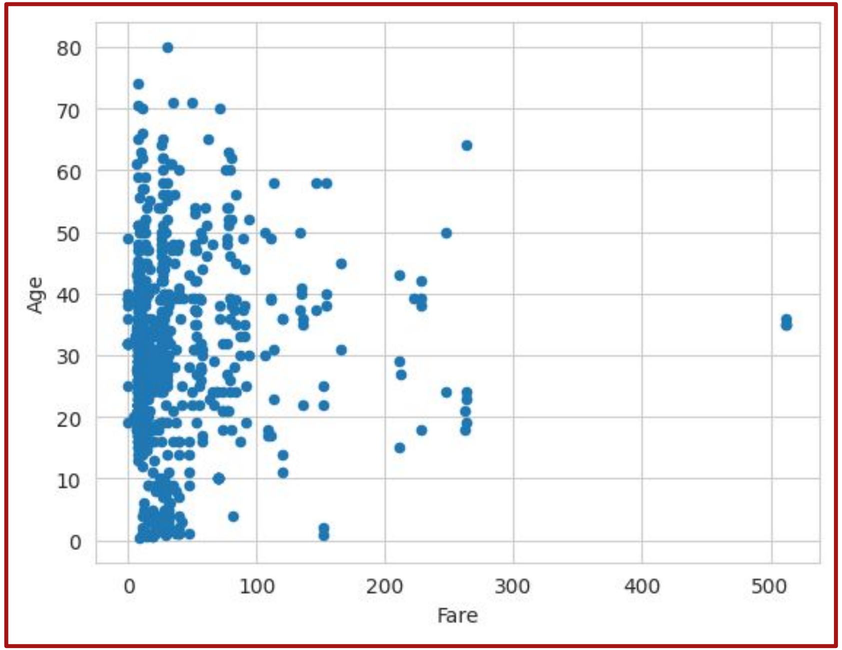
13.4.2

```
df.plot(kind='scatter', x=________, y='Age')
```

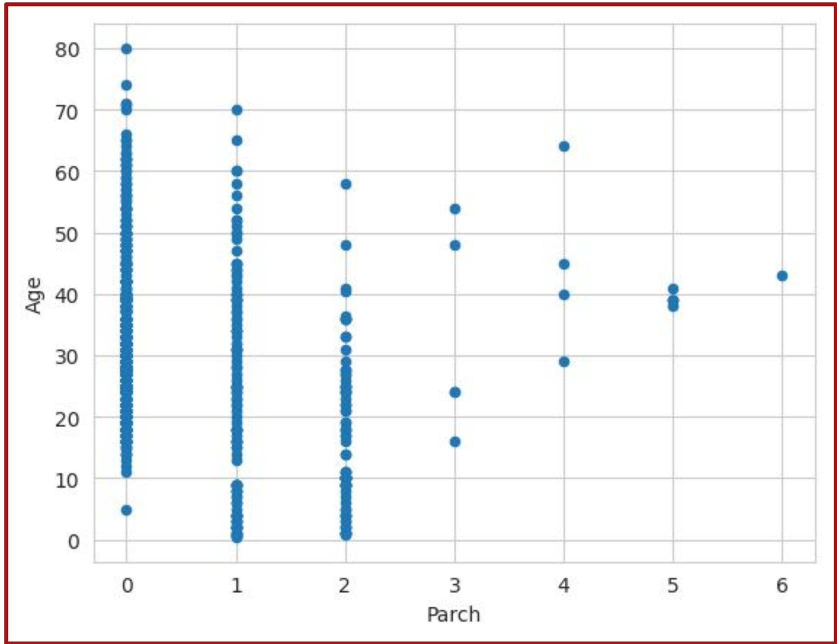
Try multiple columns which ones seem to have most graphic correlation?

Analysing the “Age” Column

Is **“Fare”** correlated?



Is **“Parch”** correlated?



Solution 13.4.2

```
df.plot(kind='scatter', x='SibSp', y='Age')
```

Analysing the “Age” Column

We can actually view the correlations across all columns in the dataframe:

13.4.3

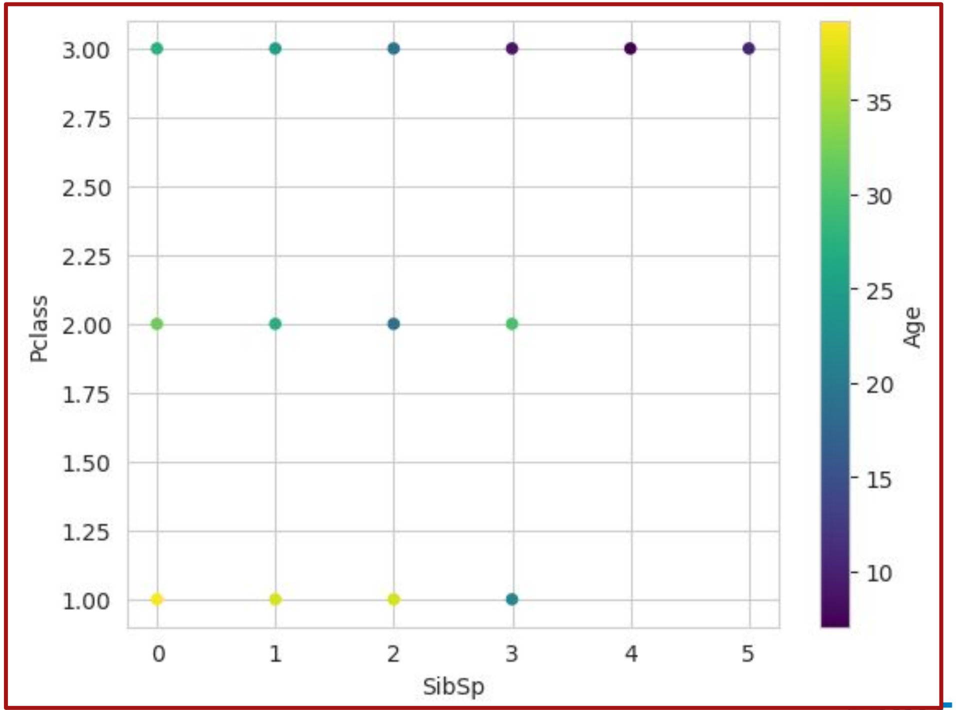
Analysing the “Age” Column

Let’s visualise how **Pclass** and **SibSp** changes affect the average Age value:

13.4.4

You can see the raw numbers:

13.4.5



Analysing the “Age” Column

We can therefore set the missing values to match the mean of the corresponding **Pclass** and **Parch** column values: **13.4.6**

But did we miss some rows? How?

13.4.7

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
159	160	0	3	Sage, Master. Thomas Henry	male	NaN	8	2	CA. 2343	69.55	NaN	S
180	181	0	3	Sage, Miss. Constance Gladys	female	NaN	8	2	CA. 2343	69.55	NaN	S
201	202	0	3	Sage, Mr. Frederick	male	NaN	8	2	CA. 2343	69.55	NaN	S
324	325	0	3	Sage, Mr. George John Jr	male	NaN	8	2	CA. 2343	69.55	NaN	S
792	793	0	3	Sage, Miss. Stella Anna	female	NaN	8	2	CA. 2343	69.55	NaN	S
846	847	0	3	Sage, Mr. Douglas Bullen	male	NaN	8	2	CA. 2343	69.55	NaN	S
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	NaN	8	2	CA. 2343	69.55	NaN	S

Analysing the “Age” Column

Exercise 13.4.8: We need to use the table we constructed in exercise **13.4.5** to pick a sensible default for these missing rows:

13.4.8

```
df['Age'] = df['Age'].fillna(value=________)
```

Verify that the value was set correctly by running cell **13.4.9**

Solution 13.4.8

```
df['Age'] = df['Age'].fillna(value=10.2)
```

```
df.groupby(['SibSp', 'Pclass'])['Age'].mean()
```

```
SibSp  Pclass
0      1         39.181416
       2         31.934220
       3         27.630201
1      1         37.414154
       2         27.363636
       3         24.912698
2      1         37.200000
       2         19.125000
       3         18.875000
3      1         22.000000
       2         30.000000
       3          8.875000
4      3          7.055556
5      3         10.200000
8      3               NaN
Name: Age, dtype: float64
```

Categorization

Data Science does **not** work well with strings.

Categorization is the act of mapping **strings to ints/floats**.

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	CatSex	CatEmbarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.2500	NaN	S	0	0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	71.2833	C85	C	1	1
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.9250	NaN	S	1	0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803	53.1000	C123	S	1	0
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450	8.0500	NaN	S	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.000000	0	0	211536	13.0000	NaN	S	0	0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.000000	0	0	112053	30.0000	B42	S	1	0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	24.912698	1	2	W./C. 6607	23.4500	NaN	S	1	0
889	890	1	1	Behr, Mr. Karl Howell	male	26.000000	0	0	111369	30.0000	C148	C	0	1
890	891	0	3	Dooley, Mr. Patrick	male	32.000000	0	0	370376	7.7500	NaN	Q	0	2

13.5.1

```
df["CatSex"] = df["Sex"].map({"male": 0, "female": 1})
df["CatEmbarked"] = df["Embarked"].map({"S": 0, "C": 1, "Q": 2})
```

Categorization

We have too many different **<u>Ages</u>**, we map them into buckets.

Exercise 13.5.2: We wish to have 5-year buckets, how many buckets do we need?

13.5.2

```
age_buckets = np.linspace(0, 80, ________)
```

```
array([ 0.,  5., 10., 15., 20., 25., 30., 35., 40., 45., 50., 55., 60.,
       65., 70., 75., 80.])
```

Solution 13.5.2

```
age_buckets = np.linspace(0, 80, 17)
```

```
array([ 0.,  5., 10., 15., 20., 25., 30., 35., 40., 45., 50., 55., 60.,
       65., 70., 75., 80.])
```

Categorization

Let’s apply our categorization to the **Age** column values, by creating a new column **CatAge**: **13.5.3**

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	CatSex	CatEmbarked	CatAge
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.2500	NaN	S	0	0	4
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	71.2833	C85	C	1	1	7
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.9250	NaN	S	1	0	5
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803	53.1000	C123	S	1	0	6
4	5	0	3	Allen, Mr. William Henry	male	35.000000	0	0	373450	8.0500	NaN	S	0	0	6

Categorization

We have too many different **Fares**, we map them into buckets.

Exercise 13.5.4: We wish to have 10-dollar buckets, how many buckets do we need?

13.5.4

```
fare_buckets = np.linspace(0, 520, _______)
```

```
array([  0.,  10.,  20.,  30.,  40.,  50.,  60.,  70.,  80.,  90., 100.,
       110., 120., 130., 140., 150., 160., 170., 180., 190., 200., 210.,
       220., 230., 240., 250., 260., 270., 280., 290., 300., 310., 320.,
       330., 340., 350., 360., 370., 380., 390., 400., 410., 420., 430.,
       440., 450., 460., 470., 480., 490., 500., 510., 520.])
```

Solution 13.5.4

```
fare_buckets = np.linspace(0, 520, 53)
```

```
array([  0.,  10.,  20.,  30.,  40.,  50.,  60.,  70.,  80.,  90., 100.,
       110., 120., 130., 140., 150., 160., 170., 180., 190., 200., 210.,
       220., 230., 240., 250., 260., 270., 280., 290., 300., 310., 320.,
       330., 340., 350., 360., 370., 380., 390., 400., 410., 420., 430.,
       440., 450., 460., 470., 480., 490., 500., 510., 520.])
```

Categorization

Let’s apply our categorization to the **Fare** column values, by creating a new column **CatFare**: **13.5.5**

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	CatSex	CatEmbarked	CatAge	CatFare
0	1	0	3	Braund, Mr. Owen Harris	male	22.000000	1	0	A/5 21171	7.2500	NaN	S	0	0	4	0.0
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.000000	1	0	PC 17599	71.2833	C85	C	1	1	7	7.0
2	3	1	3	Heikkinen, Miss. Laina	female	26.000000	0	0	STON/O2. 3101282	7.9250	NaN	S	1	0	5	0.0
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.000000	1	0	113803	53.1000	C123	S	1	0	6	5.0

Visualizing Correlations

Let's remove all columns we no longer need from the old dataset: **13.6.1**

All of these columns have been categorized!

	Survived	Pclass	SibSp	Parch	CatSex	CatEmbarked	CatAge	CatFare
0	0	3	1	0	0	0	4	0.0
1	1	1	1	0	1	1	7	7.0
2	1	3	0	0	1	0	5	0.0
3	1	1	1	0	1	0	6	5.0
4	0	3	0	0	0	0	6	0.0
...	...	...	...	...	...	...	...	...
886	0	2	0	0	0	0	5	1.0
887	1	1	0	0	1	0	3	2.0
888	0	3	1	2	1	0	4	2.0
889	1	1	0	0	0	1	5	2.0
890	0	3	0	0	0	2	6	0.0

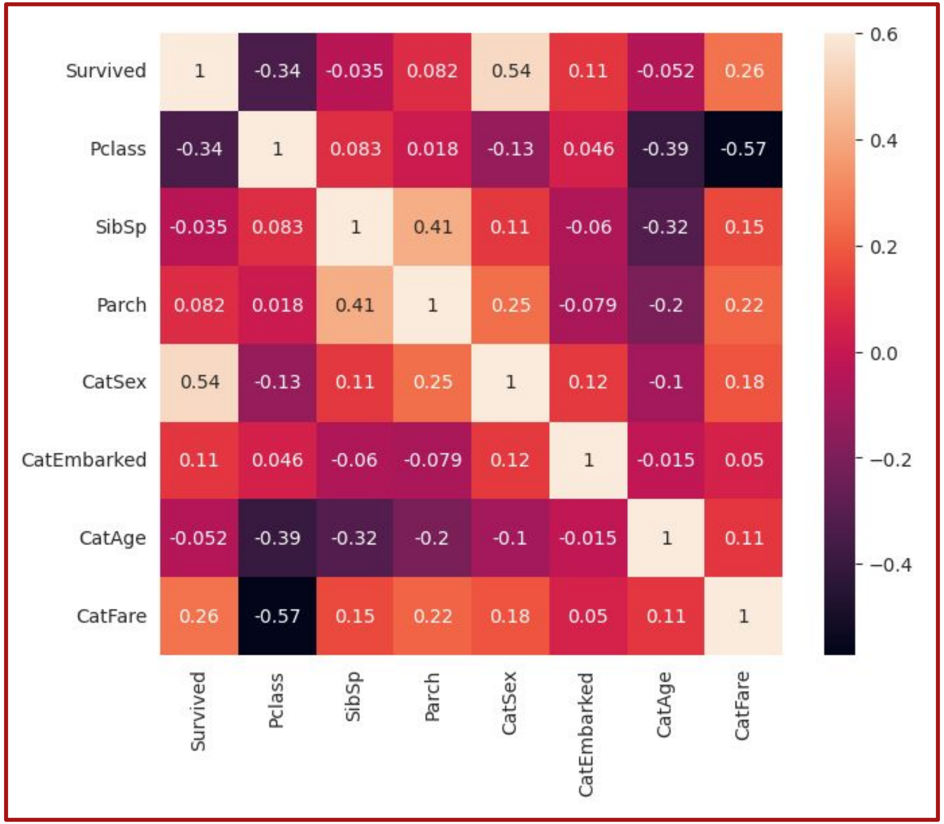
Visualizing Correlations

Understanding how **features relate to each other** is key in **prediction**.

We'll use **heatmaps** and custom plots to **visualize correlations** and interactions between variables.

Let’s see how our categorized columns correlate to the **survival of passengers**: **13.6.2**

Visualizing Correlations



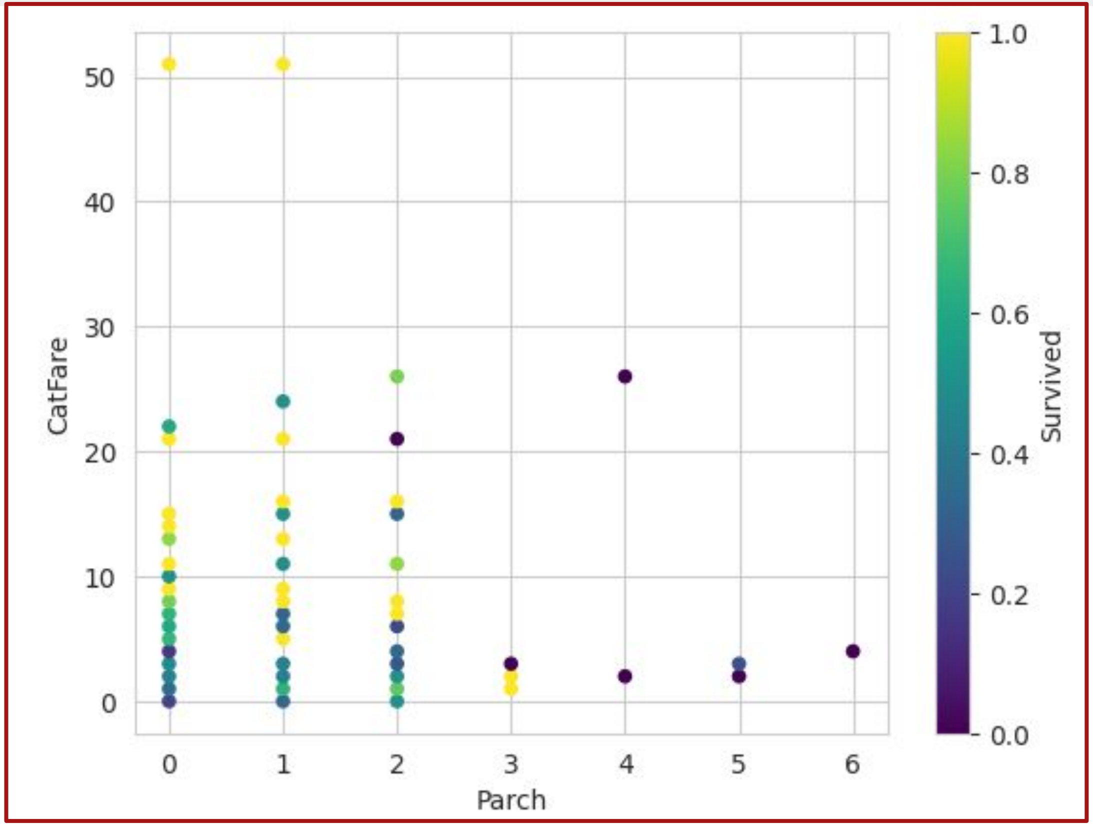
Visualizing Correlations

Look at charts produced by **13.6.3**, **13.6.4** and **13.6.5**.

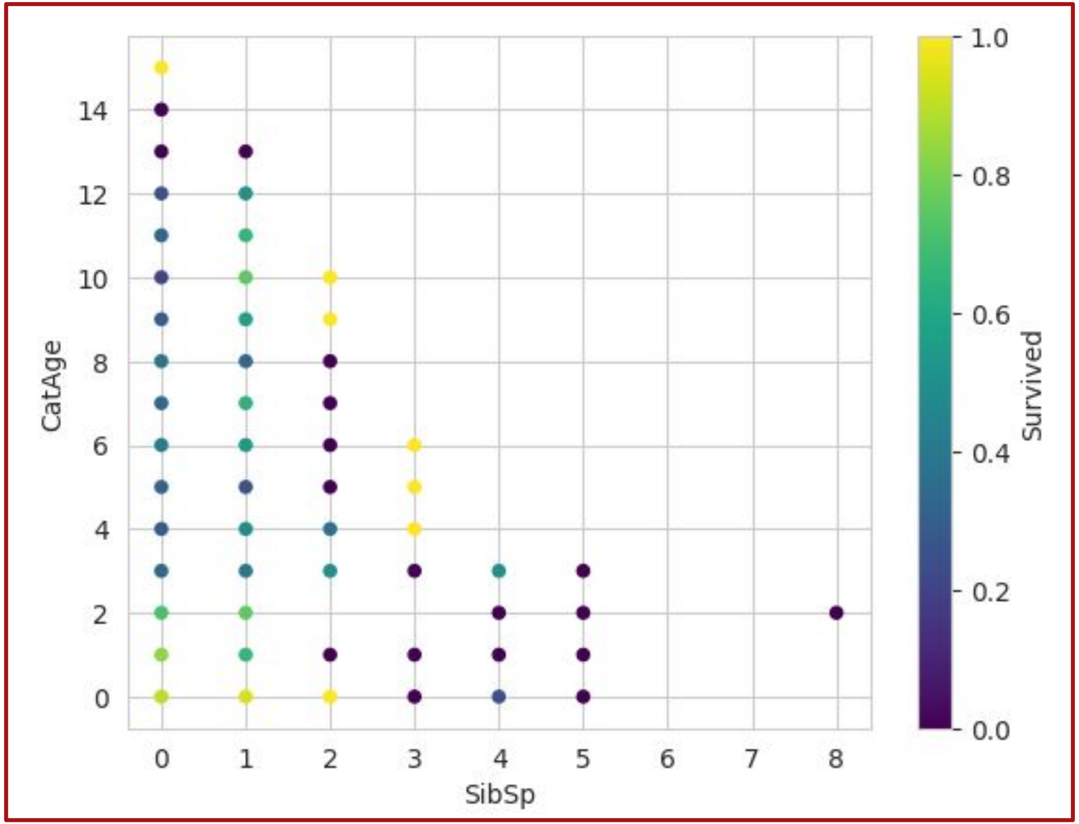
Think about what these charts are conveying.

What chart is providing **most information** with regards to the **survival rate** of passengers?

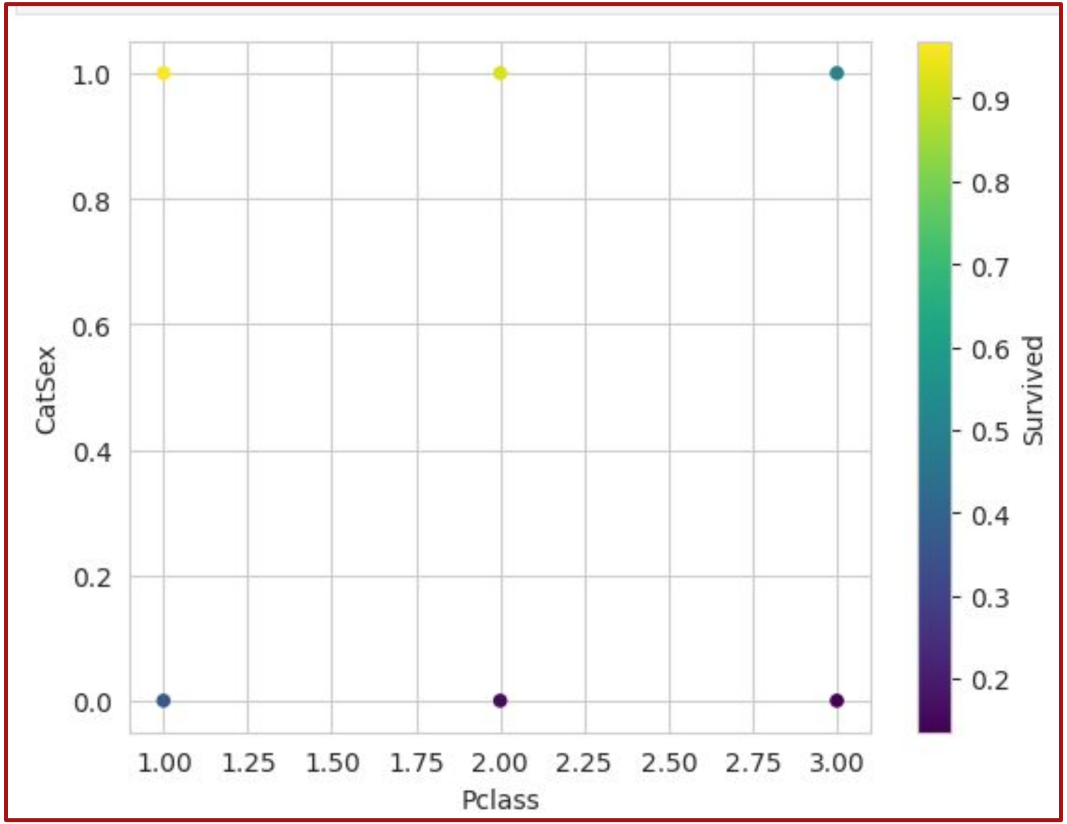
Solution 13.6.3



Solution 13.6.4



Solution 13.6.5



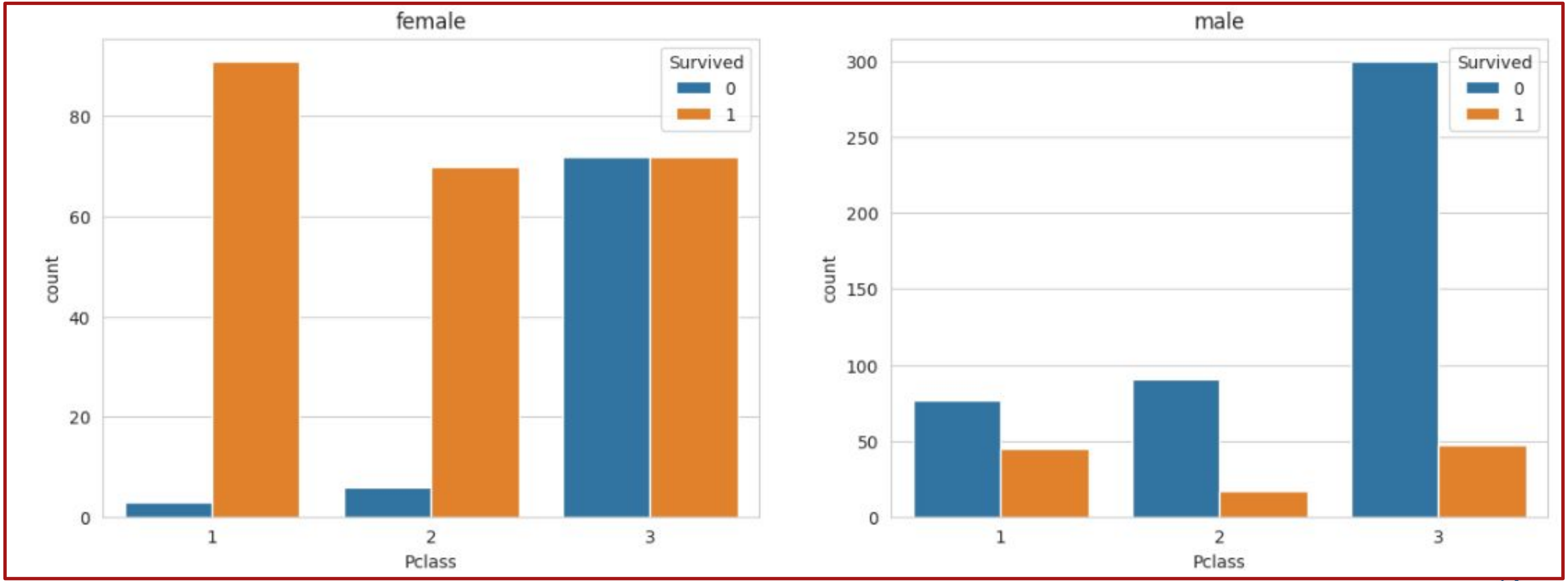
Visualizing Correlations

Is chart **13.6.5** providing us with enough information?

Let's dig deeper into the data. **13.6.6**

What is chart **13.6.6** showing us?

Solution 13.6.6



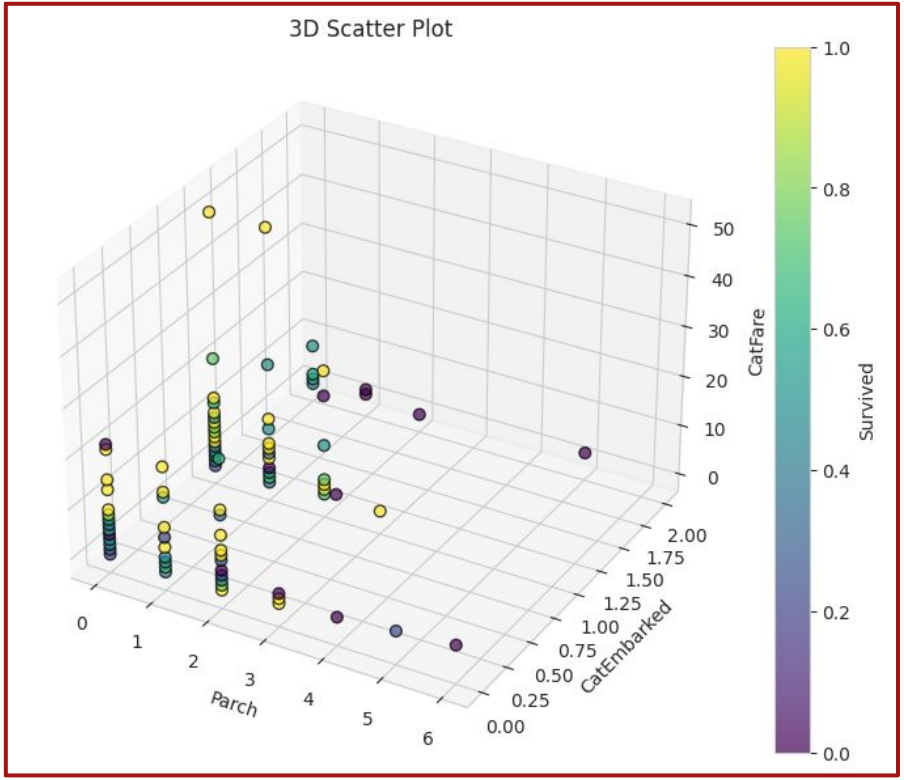
Visualizing Correlations

Look at charts produced by **13.6.7** and **13.6.8**.

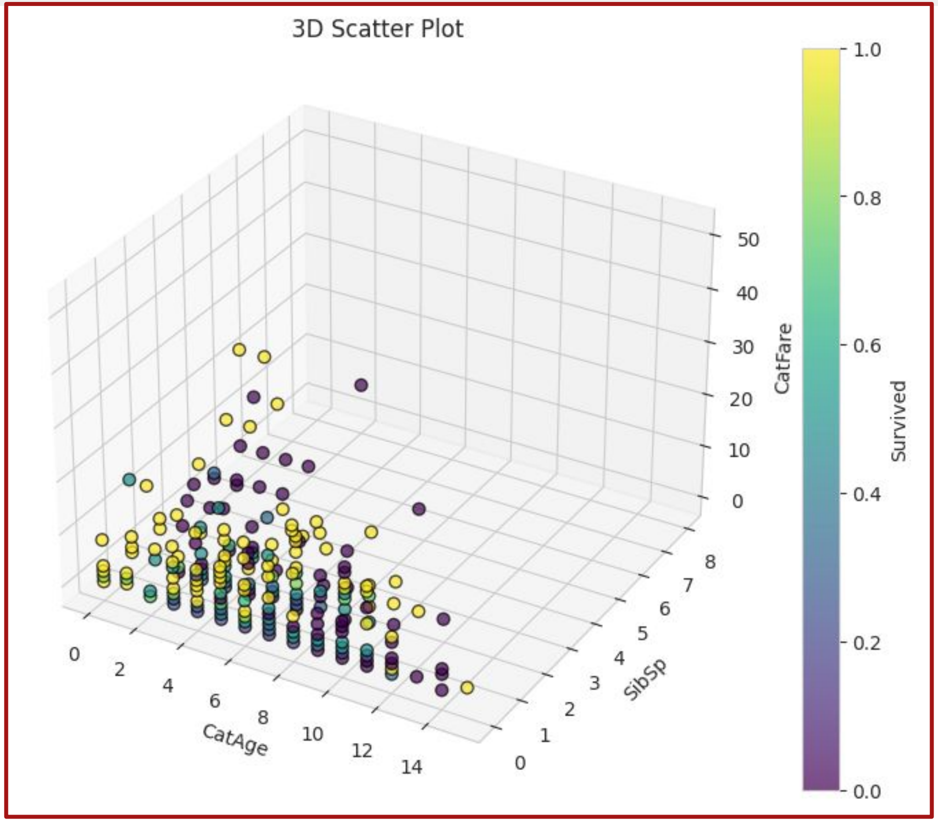
Do this charts provide meaningful correlation information?

If not try to determine a set of 3 columns in **Exercise 13.6.9** that can be used to better classify the survival rate of passengers.

Solution 13.6.7

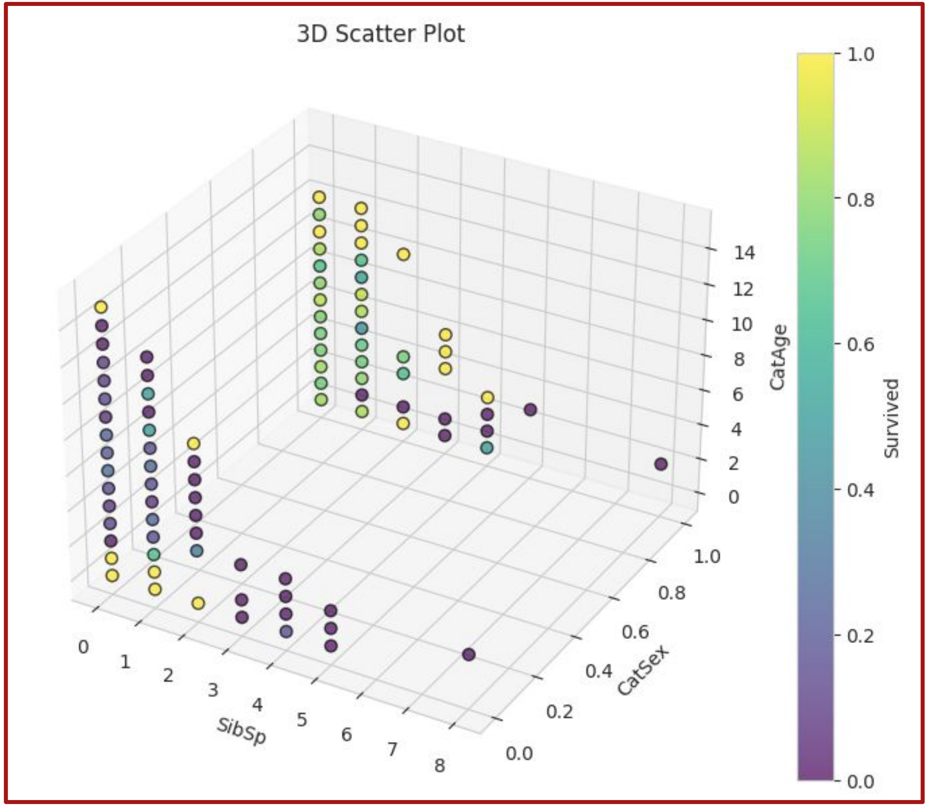


Solution 13.6.8



Solution 13.6.5

```
plot_3d("SibSp", "CatSex", "CatAge")
```



Quiz Time!

ahaslides.com/2ALVN

End of Class

See you all next week!