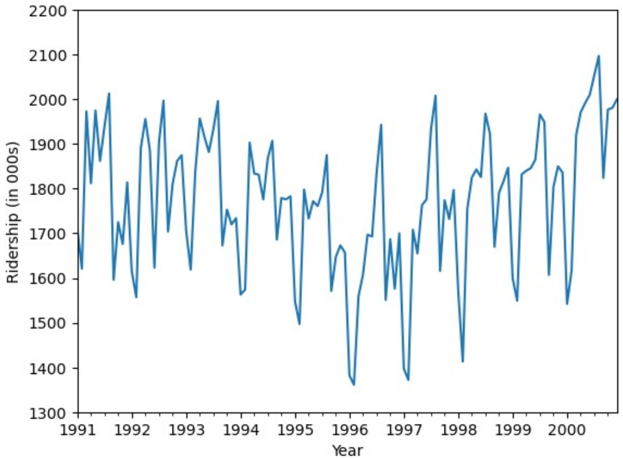
Time Series Data

	Month	Ridership	log_rider
0	Jan-91	1709	3.232742
1	Feb-91	1621	3.209783
2	Mar-91	1973	3.295127
3	Apr-91	1812	3.258158
4	May-91	1975	3.295567

```
# Time series
## Load the Amtrak data and convert them to be suitable for time series analysis
Amtrak_df = pd.read_csv(r"C:\Users\pxufre\OneDrive - Nova SBE\Ambiente de Trabalho\2957 - ABA\Amtrak.csv")
Amtrak_df.head()
```

Line graph




```
Amtrak_df['Date'] = pd.to_datetime(Amtrak_df.Month, format='%b-%y') # if date 01-1991 then format = '%m-%Y'
ridership_ts = pd.Series(Amtrak_df.Ridership.values, index=Amtrak_df.Date)

print('max: ',ridership_ts.max())
print('min: ',ridership_ts.min())
```

```
max:  2097
min:  1361
```

```
## line graph
ridership_ts.plot(ylim=[1300, 2200], legend=False)
plt.xlabel('Year')  # set x-axis label
plt.ylabel('Ridership (in 000s)')  # set y-axis label
```

The whole purpose of time series graphs is to detect historical **patterns** in the data.

DATA ANALYSIS

Finding Relationships among Variables

Relationships among Categorical Variables

```
pd.crosstab(housing_df.chas, housing_df.CAT_MEDV, margins = True)
```

Use a crosstabs to discover relationships between two categorical variables.

CAT_MEDV	0	1	All
chas			
0	398	73	471
1	24	11	35
All	422	84	506

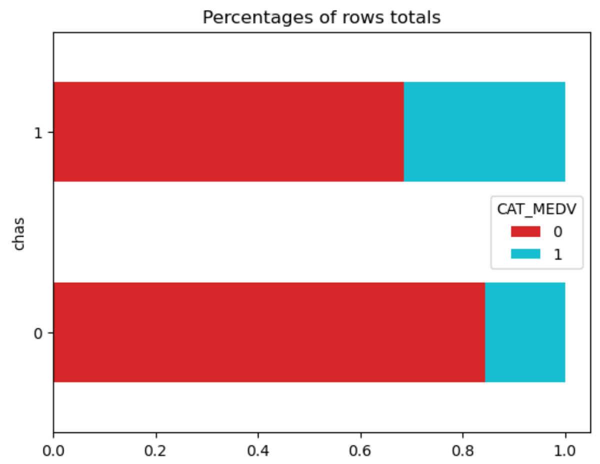
PANDAS METHOD

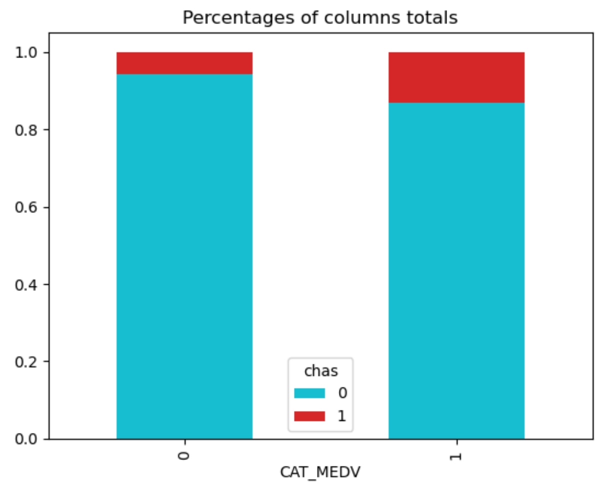
CROSSTAB

Boston Housing

CAT_MEDV	0	1
chas		
0	0.85	0.15
1	0.69	0.31

CAT_MEDV	0	1
chas		
0	0.94	0.87
1	0.06	0.13





Boston Housing

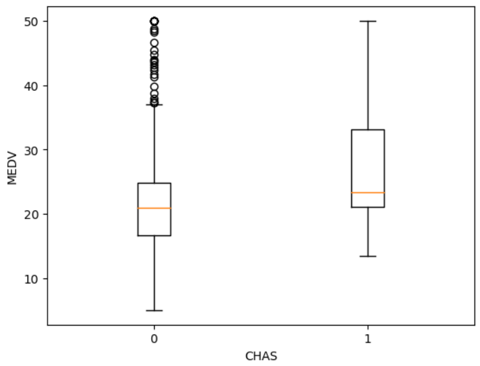
```
# create bins of size 1
housing_df['RM_bin'] = pd.cut(housing_df.rm, range(0, 10), labels=False)

# use pivot_table() to reshape data and generate pivot table
pd.pivot_table(housing_df, values='medv', index=['RM_bin'], columns=['chas'],
               aggfunc=np.mean, margins=True)
```

chas	0	1	All
RM_bin			
3	25.30	NaN	25.30
4	15.41	NaN	15.41
5	17.20	22.22	17.55
6	21.77	25.92	22.02
7	35.96	44.07	36.92
8	45.70	35.95	44.20
All	22.09	28.44	22.53

PANDAS METHOD

.PIVOT_TABLE

Relationships among Categorical Variables and a Numerical Variable



The comparison problem is one of the most important problems faced by data analysts.

```
In [108]: housing_df.groupby('chas')['medv'].describe()
```

Out[108]:

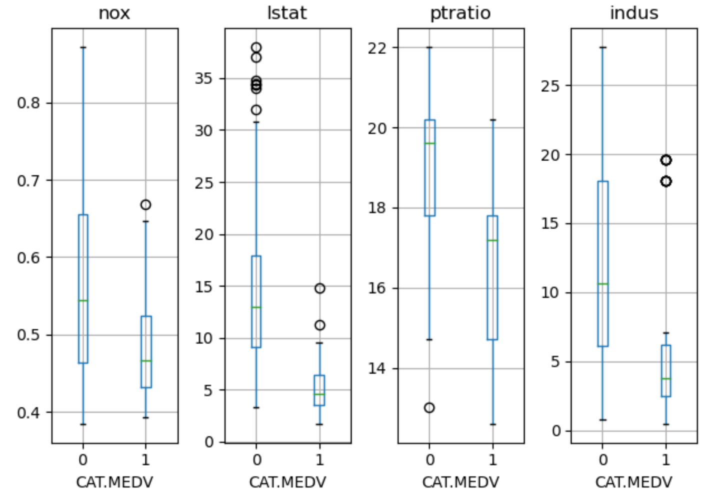
chas	count	mean	std	min	25%	50%	75%	max
0	471.00	22.09	8.83	5.00	16.60	20.90	24.80	50.00
1	35.00	28.44	11.82	13.40	21.10	23.30	33.15	50.00

PANDAS TIP

.GROUPBY()

Boston Housing

```
# side-by-side boxplots
fig, axes = plt.subplots(nrows=1, ncols=4)
housing_df.boxplot(column='nox', by='CAT_MEDV', ax=axes[0])
housing_df.boxplot(column='lstat', by='CAT_MEDV', ax=axes[1])
housing_df.boxplot(column='ptratio', by='CAT_MEDV', ax=axes[2])
housing_df.boxplot(column='indus', by='CAT_MEDV', ax=axes[3])
for ax in axes:
    ax.set_xlabel('CAT.MEDV')
plt.suptitle('')  # Suppress the overall title
plt.tight_layout()  # Increase the separation between the plots
```



Relationships among Numerical Variables

Let X_i and Y_i be the paired values for observation i, and let n be the number of observations. The **covariance** is

$$covar(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

The covariance measures the strength and direction of a linear relationship between two numeric variables.

The **correlation** is

$$corr(X,Y) = \frac{covar(X,Y)}{s_X^2 s_Y^2}$$

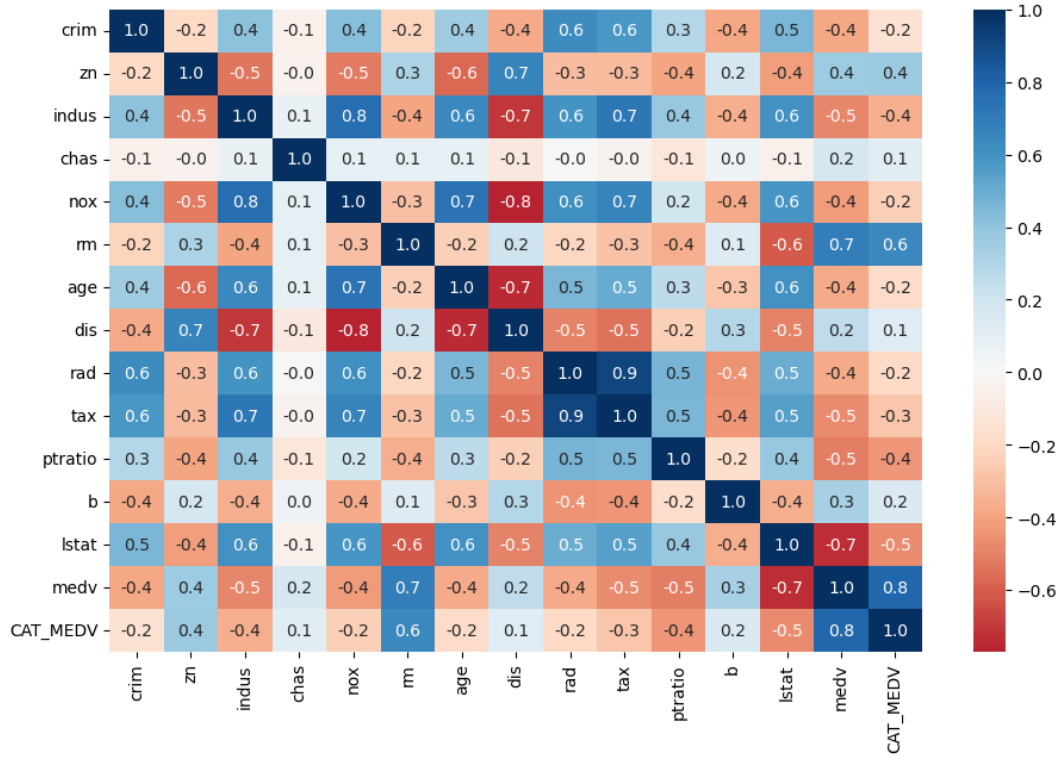
Covariance is too sensitive to the measurement scales of X and Y to make it interpretable.

$-1 \leq corr(X, Y) \leq 1$.

Boston Housing

```python
## heatmap of correlations
corr = housing_df.corr()

# Include information about values
fig, ax = plt.subplots()
fig.set_size_inches(11, 7)
sns.heatmap(corr, annot=True, fmt=".1f", cmap="RdBu", center=0, ax=ax)
```

Plot rectangular data as a color-encoded matrix.



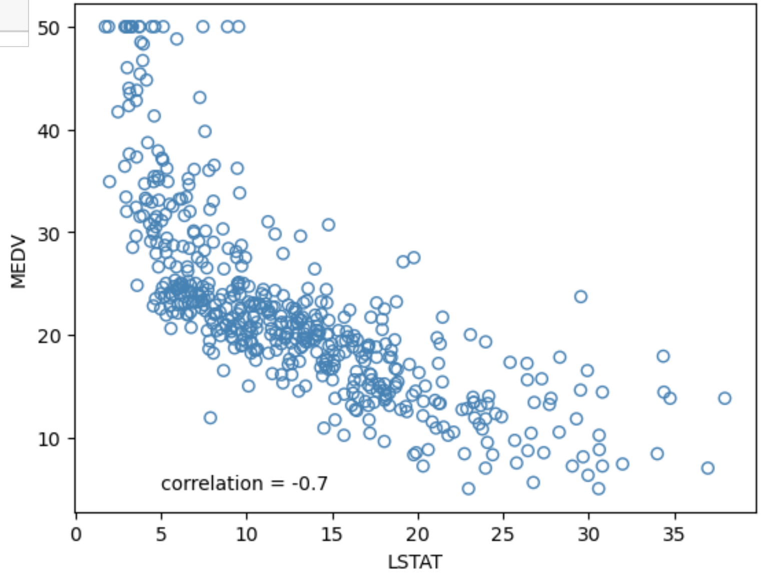
PANDAS METHOD

.CORR()

Boston Housing

```
## Set the color of the points in the scatterplot and draw as open circles.
plt.scatter(housing_df.lstat, housing_df.medv, color='steelblue', facecolor='none')
plt.xlabel('LSTAT'); plt.ylabel('MEDV')
plt.text(5,5,'correlation = -0.7')
```

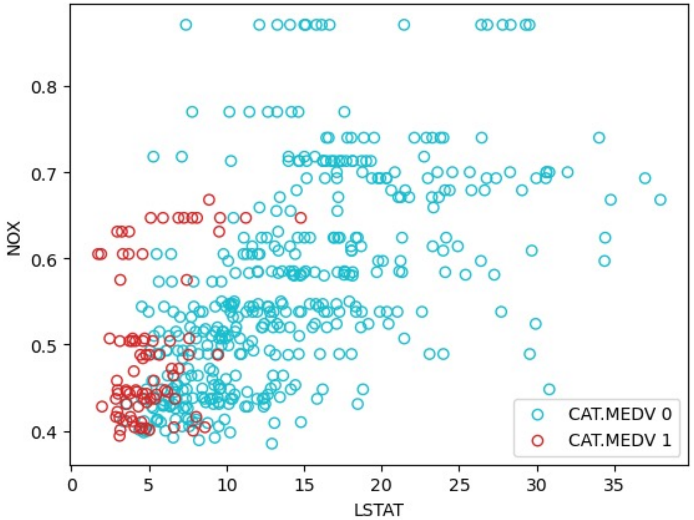
Scatter plot



DATA ANALYSIS

Multidimensional Visualization for Data Analysis

Boston Housing

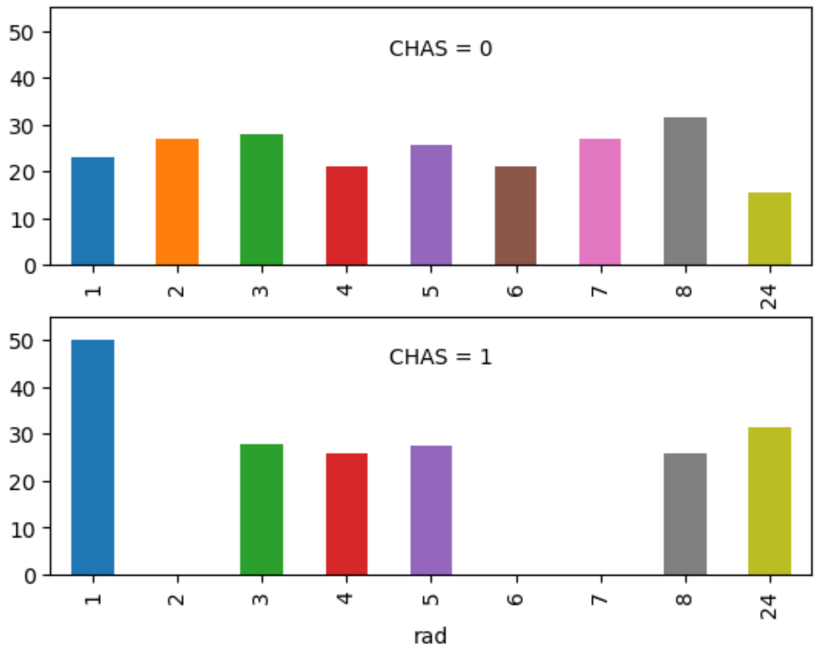



```python
# Plot first the data points for CAT_MEDV of 0 and then of 1
# Setting color to 'none' gives open circles
_, ax = plt.subplots()
for catValue, color in (0, 'C9'), (1, 'C3'):
    h_df = housing_df[housing_df.CAT_MEDV == catValue]
    ax.scatter(h_df.lstat, h_df.nox, color='none', edgecolor=color)
ax.set_xlabel('LSTAT')
ax.set_ylabel('NOX')
ax.legend(["CAT.MEDV 0", "CAT.MEDV 1"])
plt.show()
```

MATPLOTLIB

scatter

Boston Housing



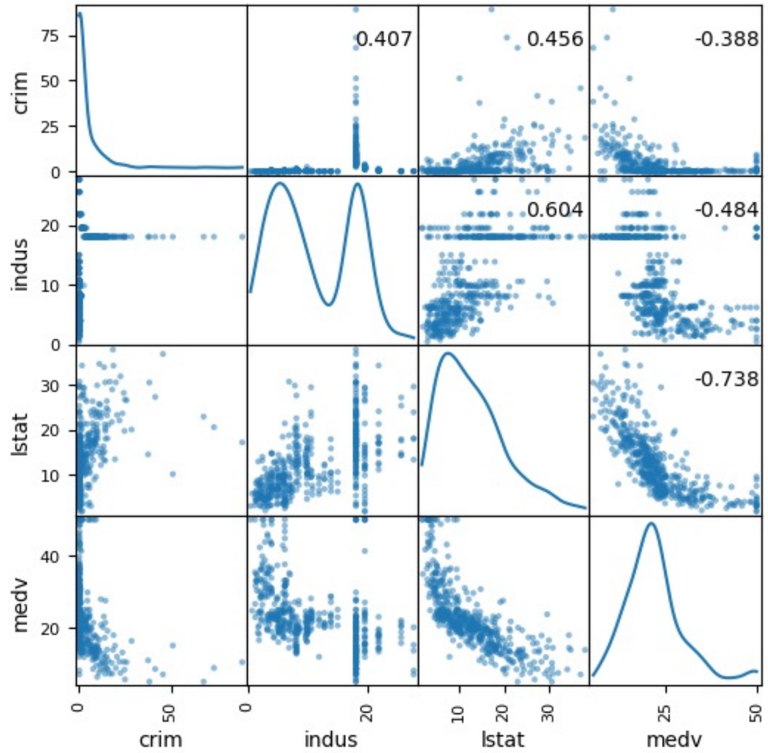

```python
## panel plots
# compute mean MEDV per RAD and CHAS
dataForPlot_df = housing_df.groupby(['chas','rad']).mean()['medv']
# We determine all possible RAD values to use as ticks
ticks = set(housing_df.rad)
for i in range(2):
    for t in ticks.difference(dataForPlot_df[i].index):
        dataForPlot_df.loc[(i, t)] = 0
# reorder to rows, so that the index is sorted
dataForPlot_df = dataForPlot_df[sorted(dataForPlot_df.index)]

colors = []
for k in range(len(ticks)):
    colors.append('C'+str(k))
# Determine a common range for the y axis
yRange = [0, max(dataForPlot_df) * 1.1]
fig, axes = plt.subplots(nrows=2, ncols=1)
dataForPlot_df[0].plot.bar(x='RAD', ax=axes[0], color= colors, ylim=yRange,)
dataForPlot_df[1].plot.bar(x='RAD', ax=axes[1], color= colors, ylim=yRange)
axes[0].annotate('CHAS = 0', xy=(3.5, 45))
axes[1].annotate('CHAS = 1', xy=(3.5, 45))
plt.show()
```

MATPLOTLIB

plot.bar

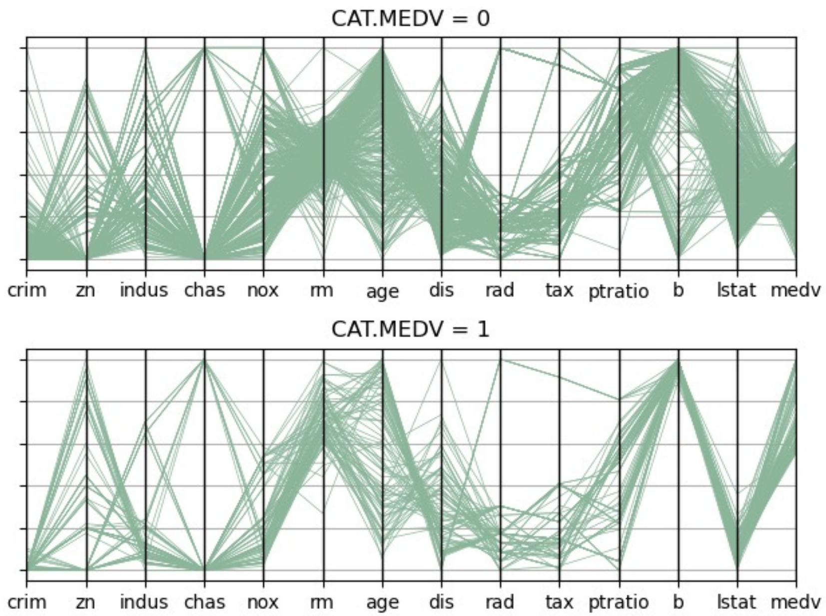
Boston Housing

```
# Display scatterplots between the different variables
# The diagonal shows the distribution for each variable
df = housing_df[['crim', 'indus', 'lstat', 'medv']]
axes = scatter_matrix(df, alpha=0.5, figsize=(6, 6), diagonal='kde')
corr = df.corr().values
for i, j in zip(*plt.np.triu_indices_from(axes, k=1)):
    axes[i, j].annotate('%.3f' %corr[i,j], (0.8, 0.8), xycoords='axes fraction', ha='center', va='center')
plt.show()
```



MATPLOTLIB
scatter_matrix

Boston Housing




```
# Transform the axes, so that they all have the same range
min_max_scaler = preprocessing.MinMaxScaler()
dataToPlot = pd.DataFrame(min_max_scaler.fit_transform(housing_df), columns=housing_df.columns)

fig, axes = plt.subplots(nrows=2, ncols=1)
for i in (0, 1):
    parallel_coordinates(dataToPlot.loc[dataToPlot.CAT_MEDV == i],
                         'CAT_MEDV', ax=axes[i], linewidth=0.5)
    axes[i].set_title('CAT.MEDV = {}'.format(i))
    axes[i].set_yticklabels([])
    axes[i].legend().set_visible(False)

plt.tight_layout()  # Increase the separation between the plots
```

MATPLOTLIB

Parallel_coordinates

Major Visualizations and Operations

Prediction

- Plot outcome on the y-axis **of boxplots, bar charts, and scatter plots**.
- Study relation of outcome to categorical predictors via **side-by-side boxplots, bar charts, and multiple panels**.
- Study relation of outcome to numerical predictors via **scatter plots**.
- Use **distribution plots** (boxplot, histogram) for determining needed transformations of the outcome variable (and/or numerical predictors).
- Examine **scatter plots with added color/panels/size** to determine the need for interaction terms.
- Use various aggregation levels and zooming to determine areas of the data with different behavior, and to evaluate the level of global vs. local patterns.

Major Visualizations and Operations

Classification

- Study relation of outcome to categorical predictors using **bar charts** with the outcome on the y-axis.
- Study relation of outcome to pairs of numerical predictors via **color-coded scatter plots** (color denotes the outcome).
- Study relation of outcome to numerical predictors via **side-by-side boxplots**: Plot boxplots of a numerical variable by outcome. Create similar displays for each numerical predictor. The most separable boxes indicate potentially useful predictors.
- Use color to represent the outcome variable on a **parallel coordinate plot**.
- Use distribution plots (**boxplot, histogram**) for determining needed transformations of numerical predictor variables.
- Examine **scatter plots** with added color/panels/size to determine the need for interaction terms.
- Use various **aggregation levels and zooming** to determine areas of the data with different behavior, and to evaluate the level of global vs. local patterns.

Major Visualizations and Operations

Time series Forecasting

- Create **line graphs** at different **temporal aggregations** to determine types of patterns.
- Use **zooming and panning** to examine various shorter periods of the series to determine areas of the data with different behavior.
- Use various **aggregation** levels to identify global and local patterns.
- Identify **missing values** in the series (that will require handling).
- Overlay **trend lines** of different types to determine adequate modeling choices.

Major Visualizations and Operations

Unsupervised Learning

- Create scatter plot matrices to identify pairwise relationships and clustering of observations.
- Use heatmaps to examine the correlation table.
- Use various aggregation levels and zooming to determine areas of the data with different behavior.
- Generate a parallel coordinates plot to identify clusters of observations.

DATA ANALYSIS

Dimensionality Reduction Techniques

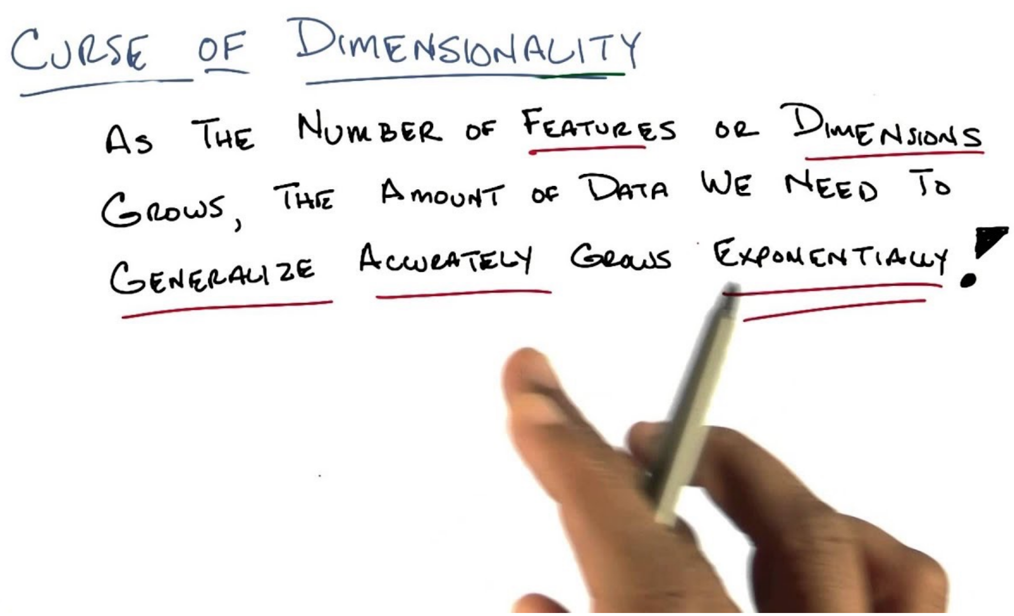
NOVA SCHOOL OF BUSINESS & ECONOMICS

PATRÍCIA XUFRE

Curse of Dimensionality

… refers to the difficulties that arise when analyzing or modeling data with many dimensions.

Data points become increasingly spread out (Data Sparsity), making it hard to find patterns or relationships.



Three simple techniques to reduce dimensionality

- **Missing values ratio**

  If the percentage of missing values in a variable exceeds the threshold (a pre-specified value), you can drop the variable.

- **Low variance filter**

  All the data columns with variance lower than the threshold value will be eliminated.

- **High correlation filter**

  All the pairs of columns having a correlation coefficient higher than the set threshold will be reduced to 1.