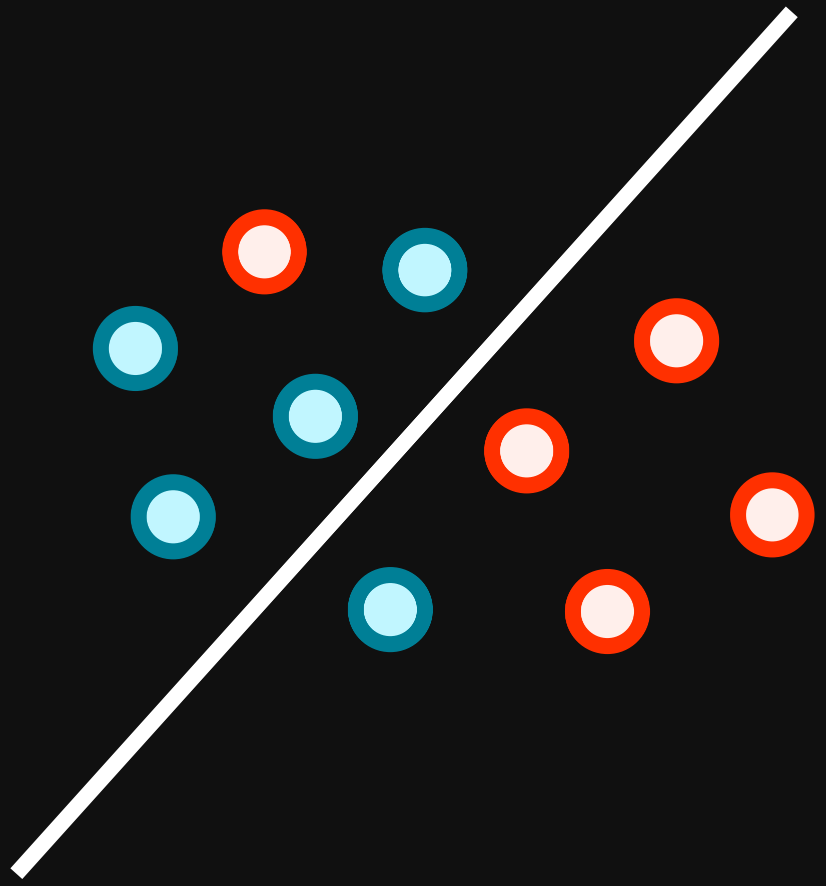
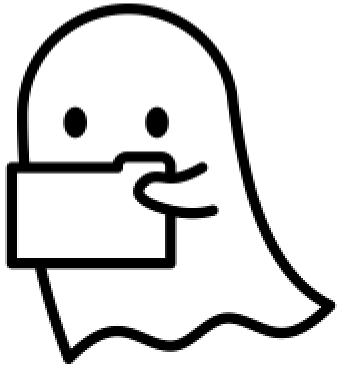
Basics of Machine Learning

Dmitry Ryabokon, github.com/dryabokon



Lesson 05
Feature Engineering



Feature Engineering

Goals

- Prepare dataset by addressing constraints/limitations of ML algorithms
- Enhance performance of ML models

Feature Engineering

Techniques

- Handling Outliers
- Imputation for Missing values
- Encoding
- Scaling: MinMax, Standard Scaler or Z Score
- Handling imbalanced data
- Dimensionality reduction

- Binning
- Log transform (Box-Cox, Yeo-Johnson)
- Grouping operations
- Feature split

Visualization

- Univariate
- Bivariate
- Multivariate

Outliers

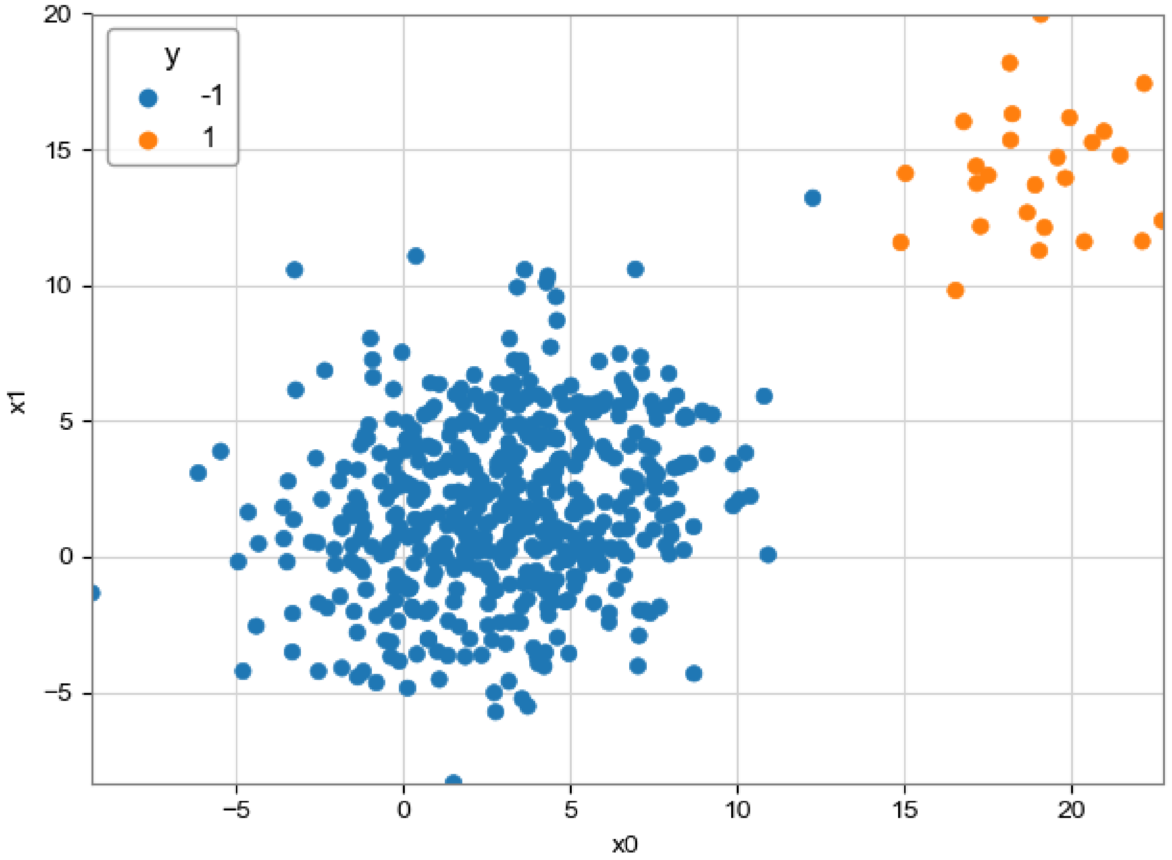
Outliers

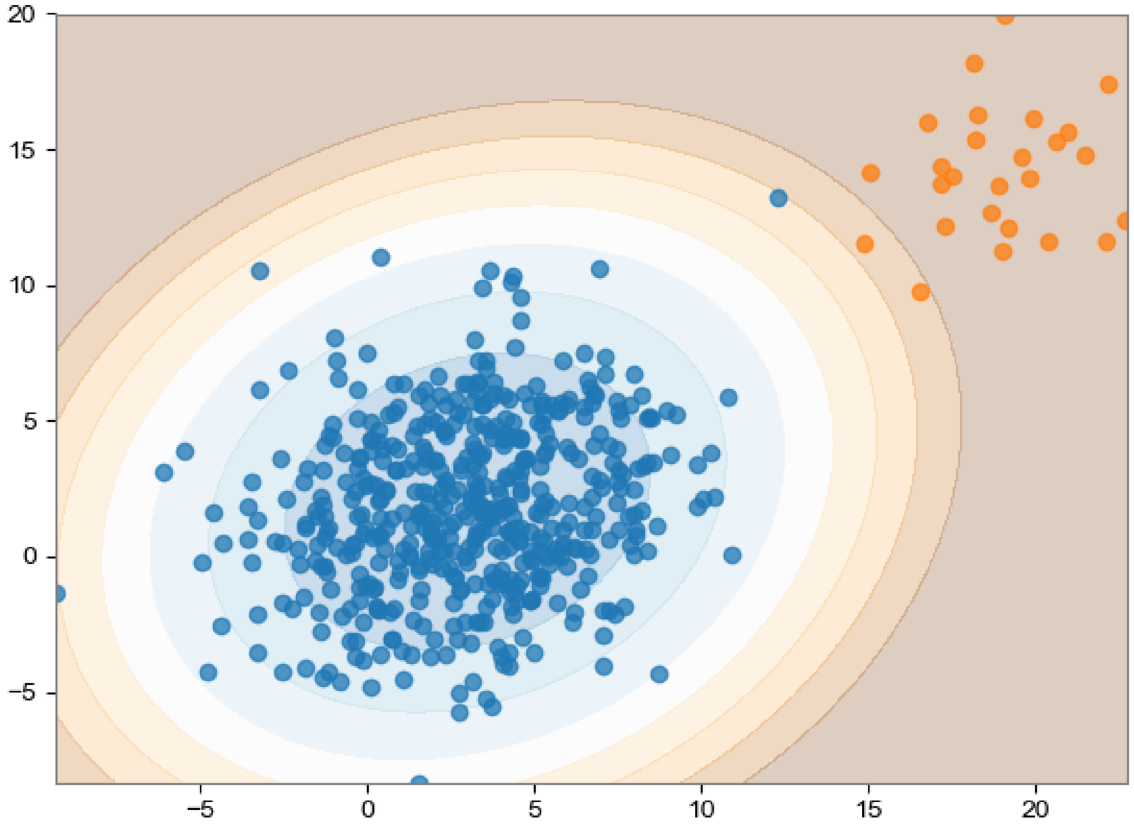
A data point that is considerably distant from the other similar data points. There are some methods to deal with outliers :

- Extreme Value Analysis
- Isolation Forest
- Minimum Covariance Determinant
- Local Outlier Factor
- One-Class SVM

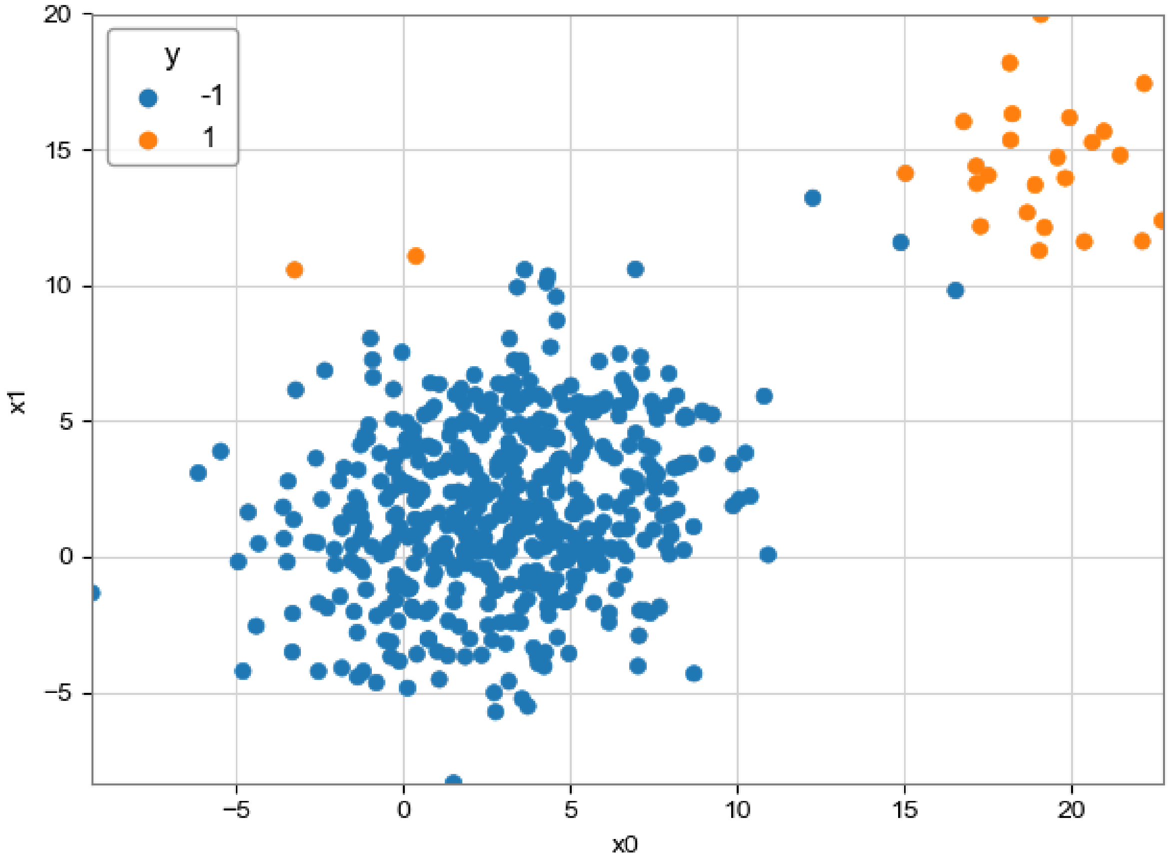
https://machinelearningmastery.com/model-based-outlier-detection-and-removal-in-python/
https://scikit-learn.org/stable/modules/outlier_detection.html

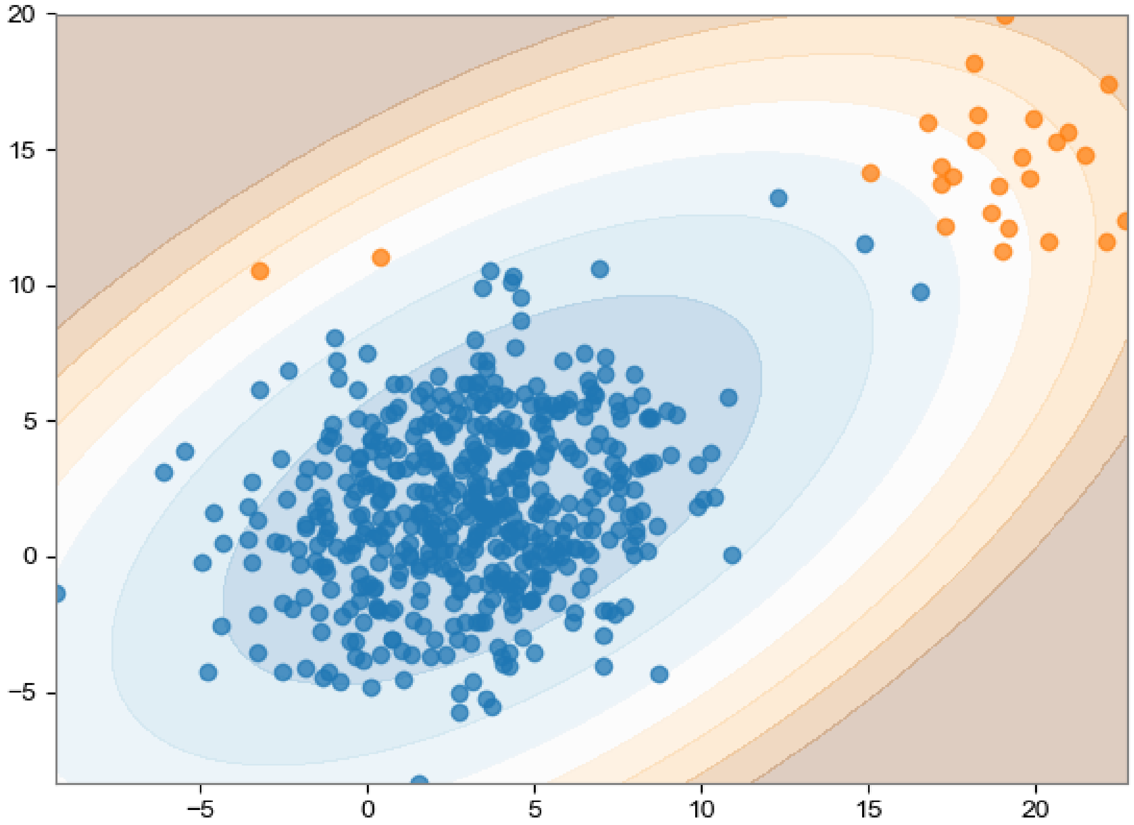
Outliers: GT



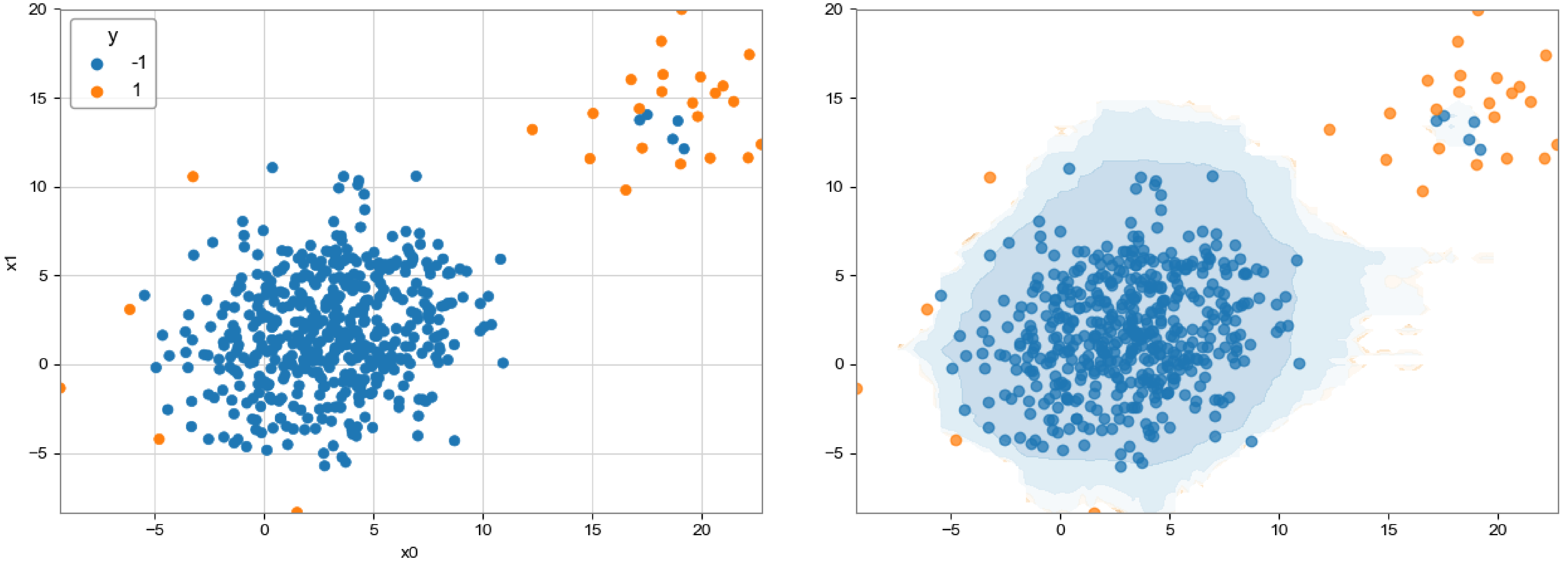


Outliers: Normal distribution estimation

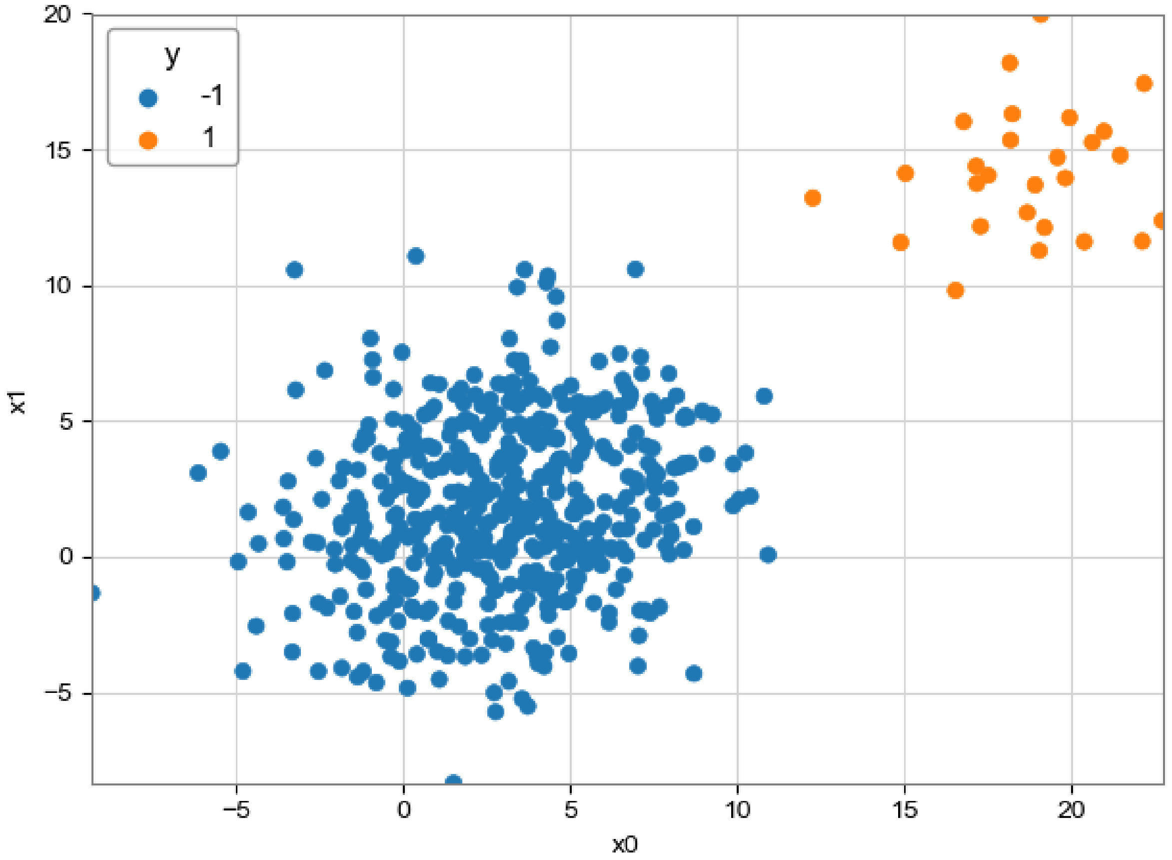


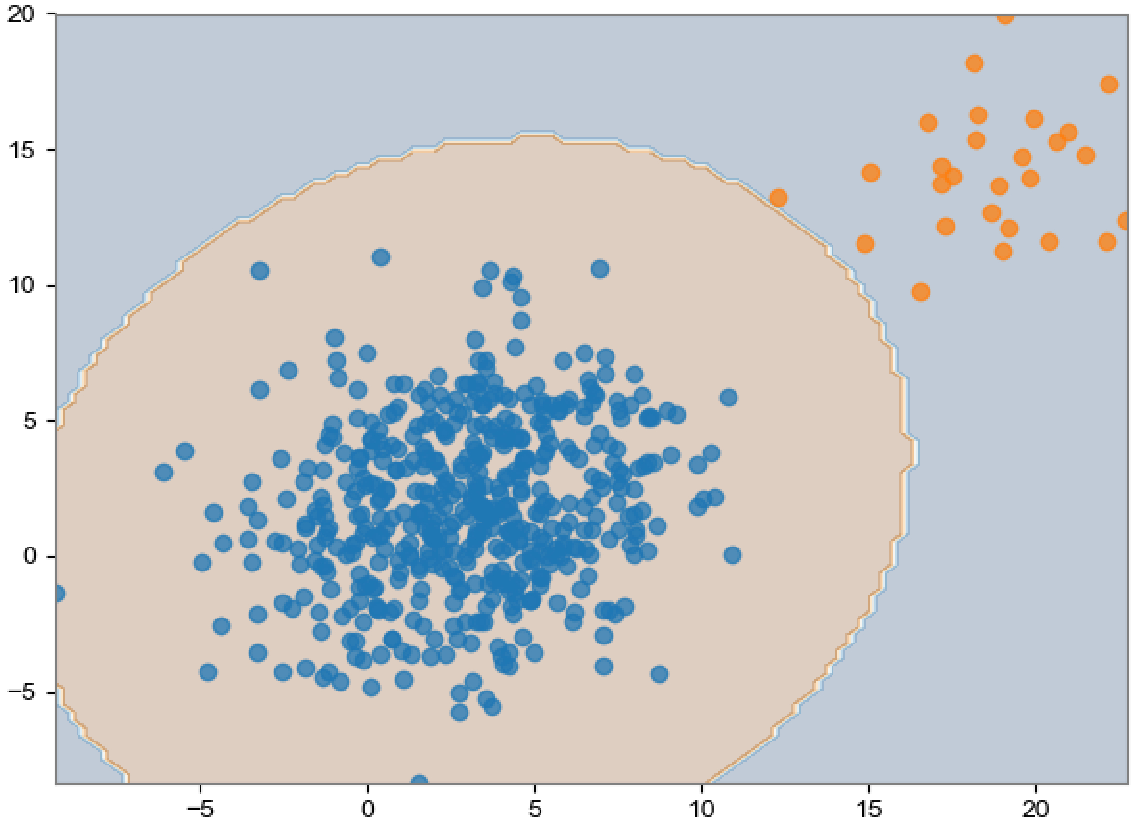


Outliers: Isolation Forest

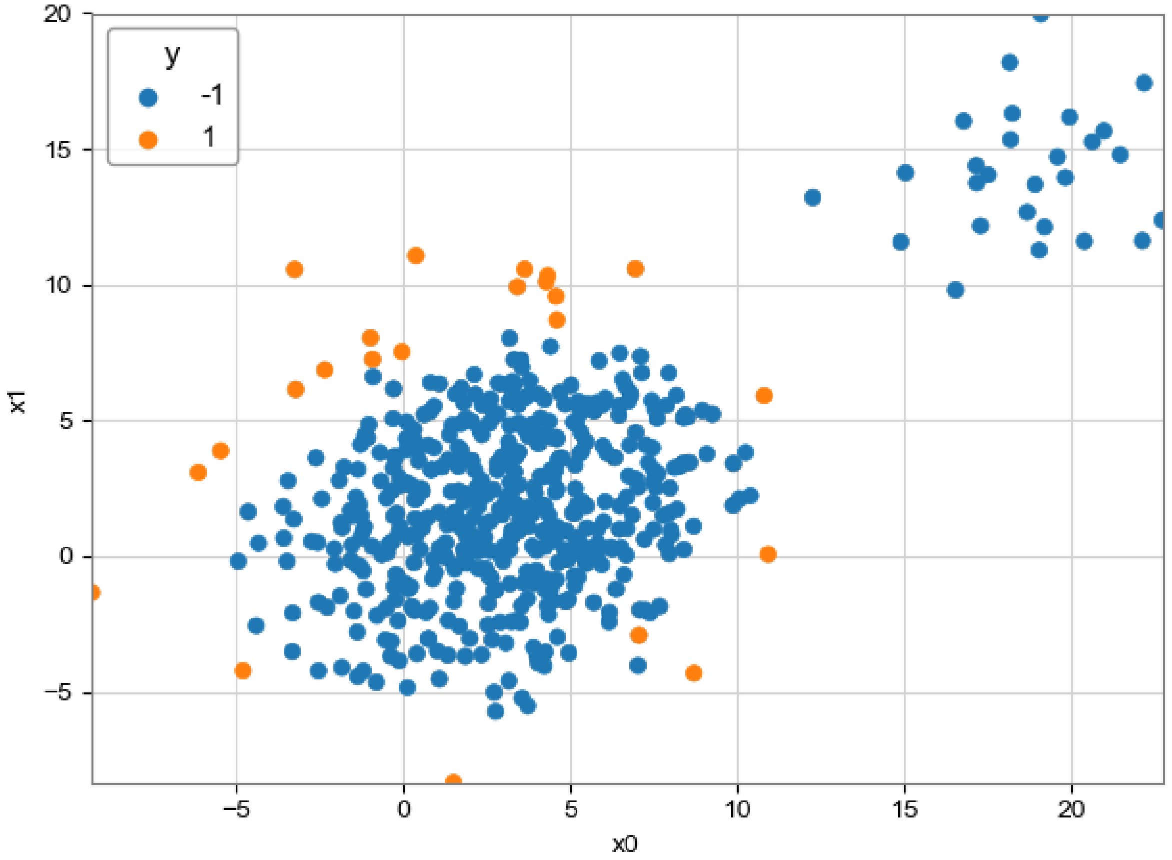


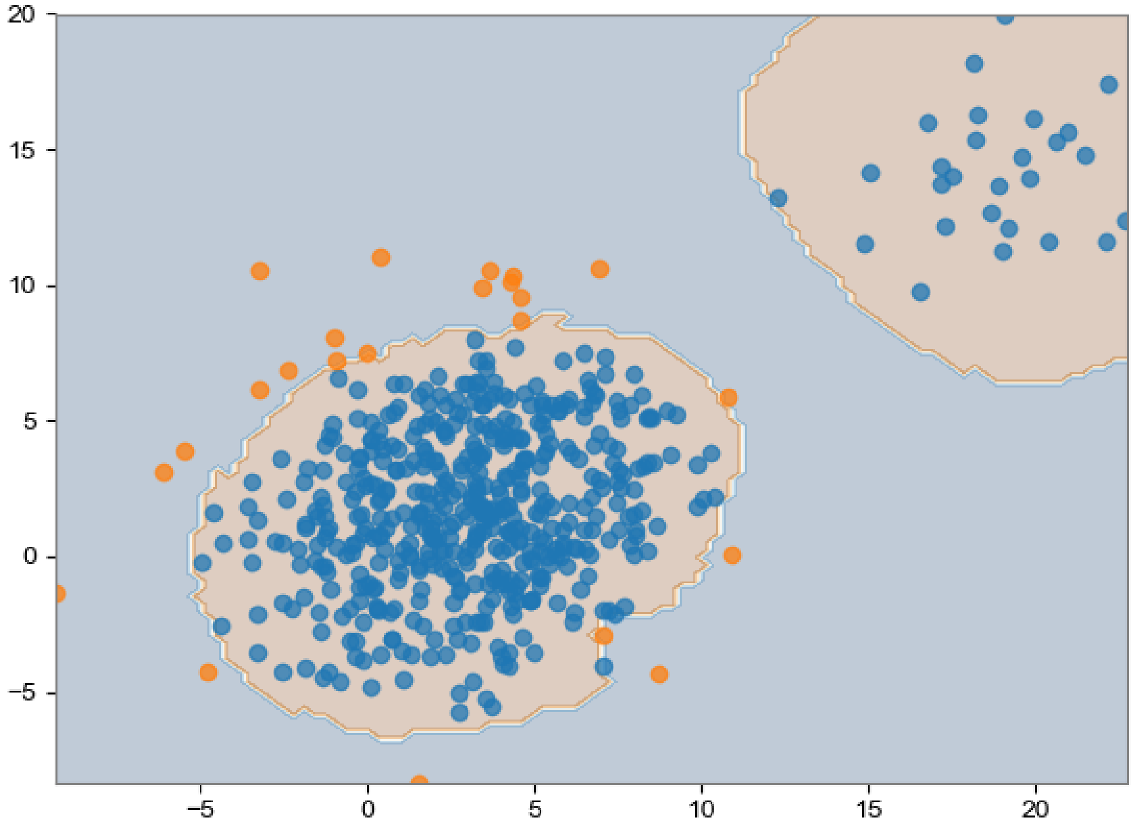
Outliers: Elastic Envelope





Outliers: Local Outlier Factor





Imputation

Imputation

Imputation is the process of replacing missing data with substituted values.
There are several ways to handle it

- Do Nothing
- Mean/Median
- Most Frequent or Constant (e.g. Zero)
- Nearest (KNN)
- Extrapolation and Interpolation
- Stochastic regression
- Encoding (e.g. HotDeck)

https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779

Missing values

```
df_dummy = pd.DataFrame(numpy.array([[5,9,-5,999,3],
                                     [7,numpy.NaN,0,1,0],
                                     [9,numpy.NaN,25,-1,numpy.NaN]]).T)
```

```
A = (df.isnull()).to_numpy()

cv2.imwrite(folder_out + 'nans_1.png', 255 * A)
print(df)
print()
print(A)
```

```
   0     1     2
0    5.0  7.0   9.0
1    9.0  NaN   NaN
2   -5.0  0.0  25.0
3  999.0  1.0  -1.0
4    3.0  0.0   NaN

[[False False False]
 [False  True  True]
 [False False False]
 [False False False]
 [False False  True]]
```

Feature Engineering

Imputation: replace

```
df_dummy = pd.DataFrame(numpy.array([[5,9,-5,999,3],
                                     [7,numpy.NaN,0,1,0],
                                     [9,numpy.NaN,25,-1,numpy.NaN]]).T)

 dct_replace = {numpy.NaN: 999.0}

 print(df)
 print()
 df.replace(dct_replace, inplace=True)
 print(df)
```

```
 0     1     2
0     5.0  7.0   9.0
1     9.0  NaN   NaN
2    -5.0  0.0  25.0
3   999.0  1.0  -1.0
4     3.0  0.0   NaN

       0      1      2
0     5.0    7.0    9.0
1     9.0  999.0  999.0
2    -5.0    0.0   25.0
3   999.0    1.0   -1.0
4     3.0    0.0  999.0
```

Feature Engineering

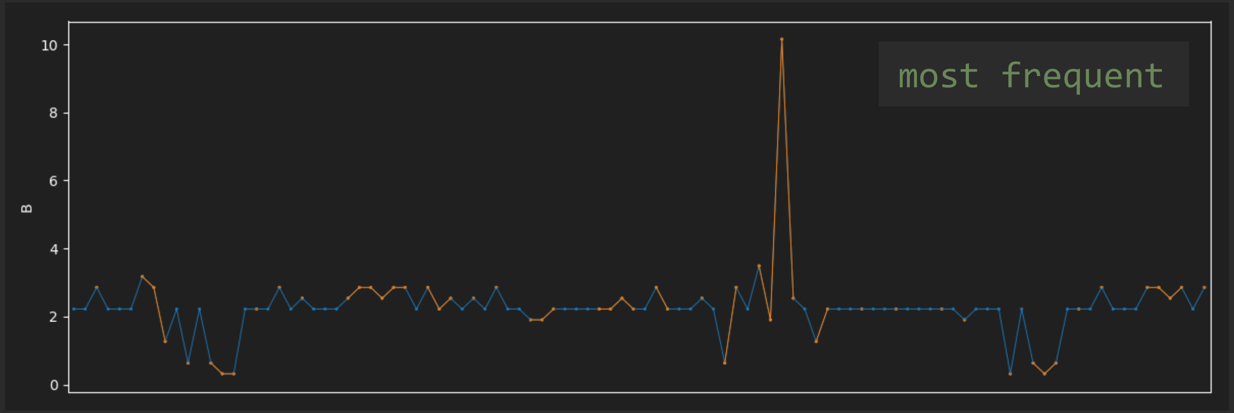
Imputation: SimpleImputer

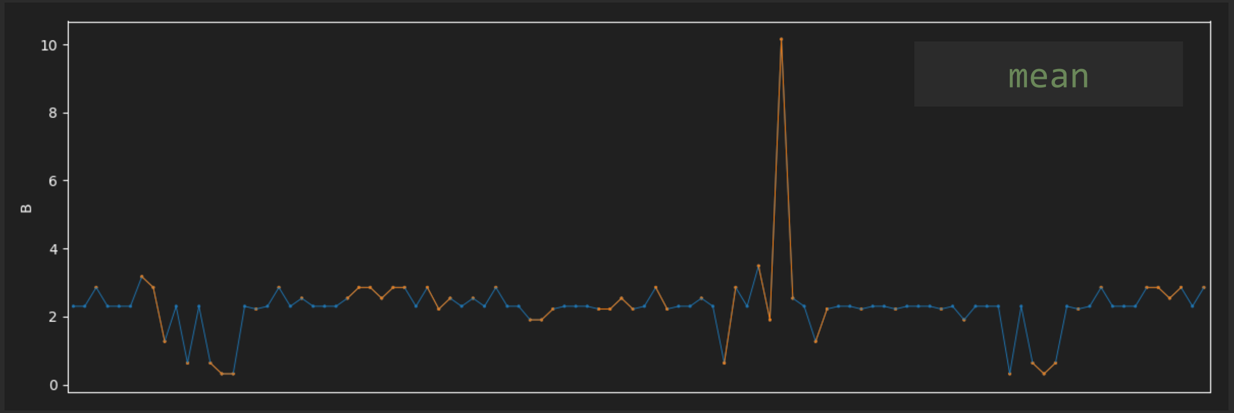
```
df_dummy = pd.DataFrame(numpy.array([[5,9,-5,999,3],
                                     [7,numpy.NaN,0,1,0],
                                     [9,numpy.NaN,25,-1,numpy.NaN]]).T)

strategies = ['mean', 'median', 'most_frequent', 'interpolate']
for strategy in strategies:
    if strategy == 'interpolate':
        B = df.iloc[:,[idx_column]].interpolate()
    else:
        imp = SimpleImputer(missing_values=numpy.nan,strategy=strategy)
        B = pd.DataFrame(data=imp.fit_transform(df.iloc[:,[idx_column]]),
                         index=df.index,columns=[df.columns[idx_column]])
```

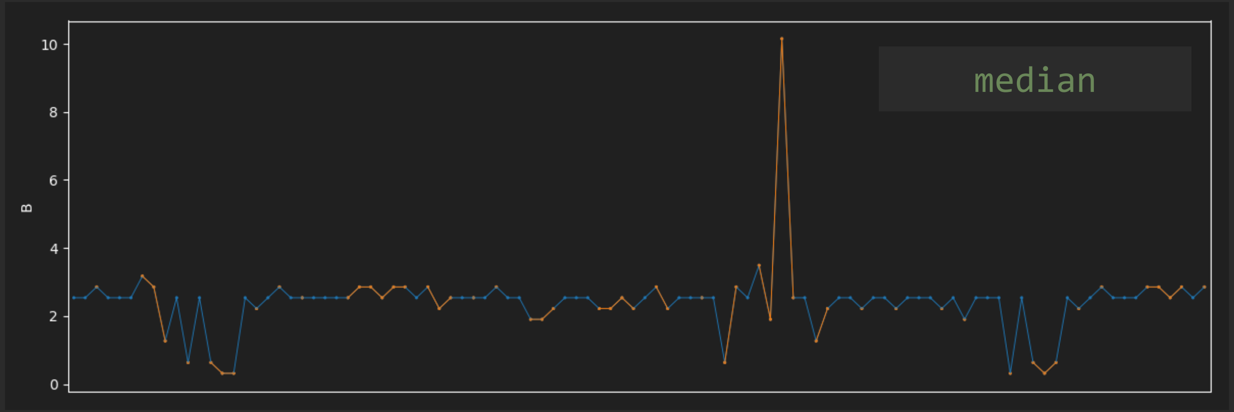
	original	mean	median	most_frequent	interpolate
0	7.0	7.0	7.0	7.0	7.0
1	NaN	2.0	0.5	0.0	3.5
2	0.0	0.0	0.0	0.0	0.0
3	1.0	1.0	1.0	1.0	1.0
4	0.0	0.0	0.0	0.0	0.0

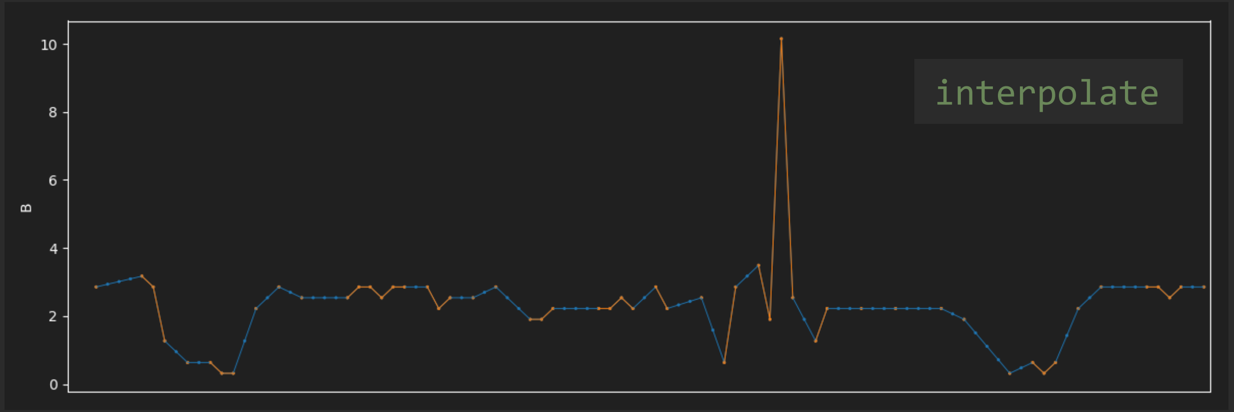
Feature Engineering

Imputation: SimpleImputer









Encoding

Feature Engineering

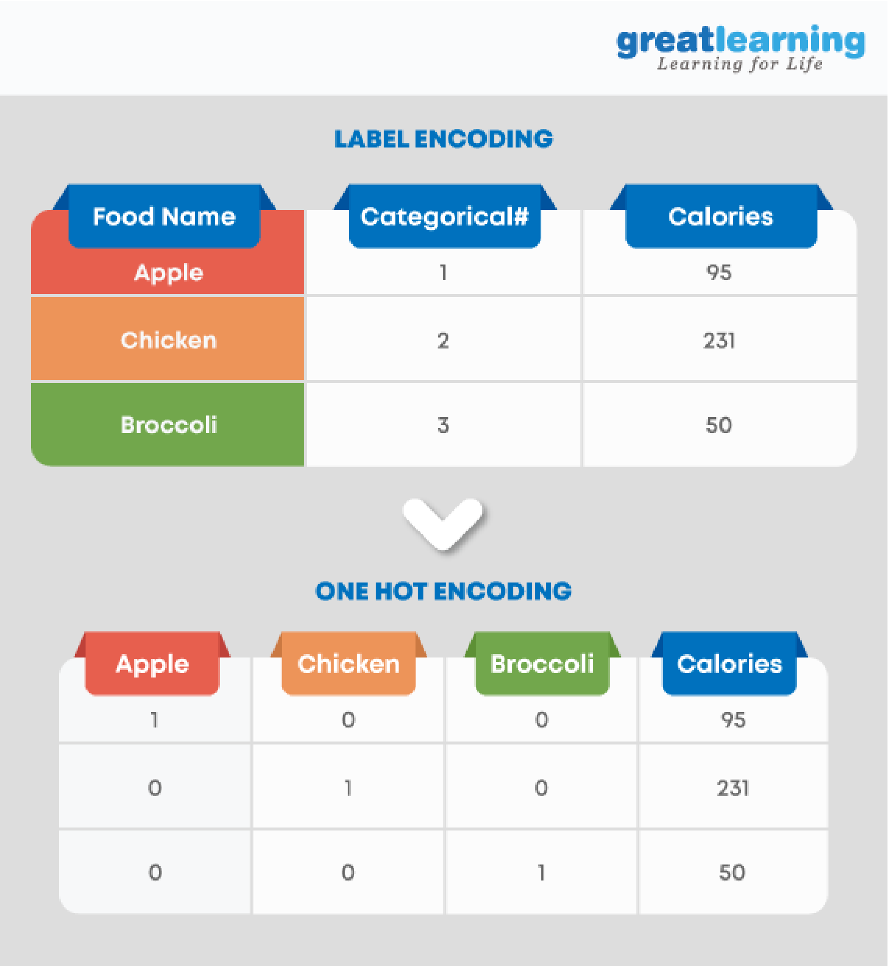
Encoding

Apply One-Hot Encoding when:

- Categorical feature is not ordinal (e.g. countries)
- small number of categorical features

Apply Label Encoding when:

- Categorical feature is ordinal (e.g. Jr. kg, Sr. kg, Primary school, high school)
- Large number of categories



LABEL ENCODING

Food Name	Categorical#	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

ONE HOT ENCODING

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

One-Hot Encoding: Dummy Variable Trap

Dummy Variable Trap is a scenario in which variables are highly correlated to each other. The outcome of one variable can easily be predicted with the help of the remaining variables. It leads to the multicollinearity issue. in order to overcome this, one of the dummy variables has to be dropped.

Feature Engineering

Ordinal Encoding:

```
df = pd.DataFrame(numpy.array([['M', 'O-', 'medium'],
                               ['M', 'O-', 'high'],
                               ['F', 'O+', 'high'],
                               ['F', 'AB', 'low'],
                               ['F', 'B+', numpy.nan]]))
df.columns = ['sex', 'blood_type', 'edu_level']
encoder = OrdinalEncoder()

print(df)
print()
df.iloc[:,2] = encoder.fit_transform(df.iloc[:, 2].values.reshape((-1, 1)))
print(df)

# drawback: missing value is encoded as a separate class
# drawback: order of data is not respected
```

```
  sex blood_type edu_level
0   M         O-    medium
1   M         O-      high
2   F         O+      high
3   F         AB       low
4   F         B+       nan

  sex blood_type  edu_level
0   M         O-        2.0
1   M         O-        0.0
2   F         O+        0.0
3   F         AB        1.0
4   F         B+        3.0
```

Feature Engineering

Ordinal Encoding: ordering

```
df = pd.DataFrame(numpy.array([['M', 'O-', 'medium'],
                               ['M', 'O-', 'high'],
                               ['F', 'O+', 'high'],
                               ['F', 'AB', 'low'],
                               ['F', 'B+', numpy.nan]]))
df.columns = ['sex', 'blood_type', 'edu_level']

print(df)
print()

cat = pd.Categorical(df['edu_level'],
                     categories=['missing', 'low', 'medium', 'high'], ordered=True)
cat.fillna('missing')

labels, unique = pd.factorize(cat, sort=True)
df.edu_level = labels
print(df)
```

```
  sex blood_type edu_level
0   M         O-    medium
1   M         O-      high
2   F         O+      high
3   F         AB       low
4   F         B+       nan

  sex blood_type  edu_level
0   M         O-          1
1   M         O-          2
2   F         O+          2
3   F         AB          0
4   F         B+         -1
```

Feature Engineering

OneHot Encoding:

```
df = pd.DataFrame(numpy.array([['M', 'O-'],
                               ['M', 'O-'],
                               ['F', 'O+'],
                               ['F', 'AB'],
                               ['F', 'B+']]))
df.columns = ['sex', 'blood_type']
onehot = OneHotEncoder(dtype=numpy.int, sparse=True)

df2 = pd.DataFrame(onehot.fit_transform(df[['sex', 'blood_type']]).toarray())
df2.columns = numpy.unique(df['sex']).tolist() + numpy.unique(df['blood_type']).tolist()


print(df.to_string(index=False))
print()
print(df2.to_string(index=False))
```

```
sex blood_type
  M         O-
  M         O-
  F         O+
  F         AB
  F         B+

 F  M  AB  B+  O+  O-
 0  1   0   0   0   1
 0  1   0   0   0   1
 1  0   0   0   1   0
 1  0   1   0   0   0
 1  0   0   1   0   0
```

Imbalanced data

Imbalanced dataset: general approaches

- Using class weights
- Collect more data to even the imbalances in the dataset
- Sampling: resample the dataset to correct for imbalances
- Choosing loss functions like Focal Loss
- Try a different algorithm altogether on your dataset

Imbalanced dataset: Sampling

- **Under Sampling:** delete or select a subset of examples from the majority class
- **Over Sampling -** up-sample the Minority class and thus solve the problem of information loss, however, we get into the trouble of having Overfitting.

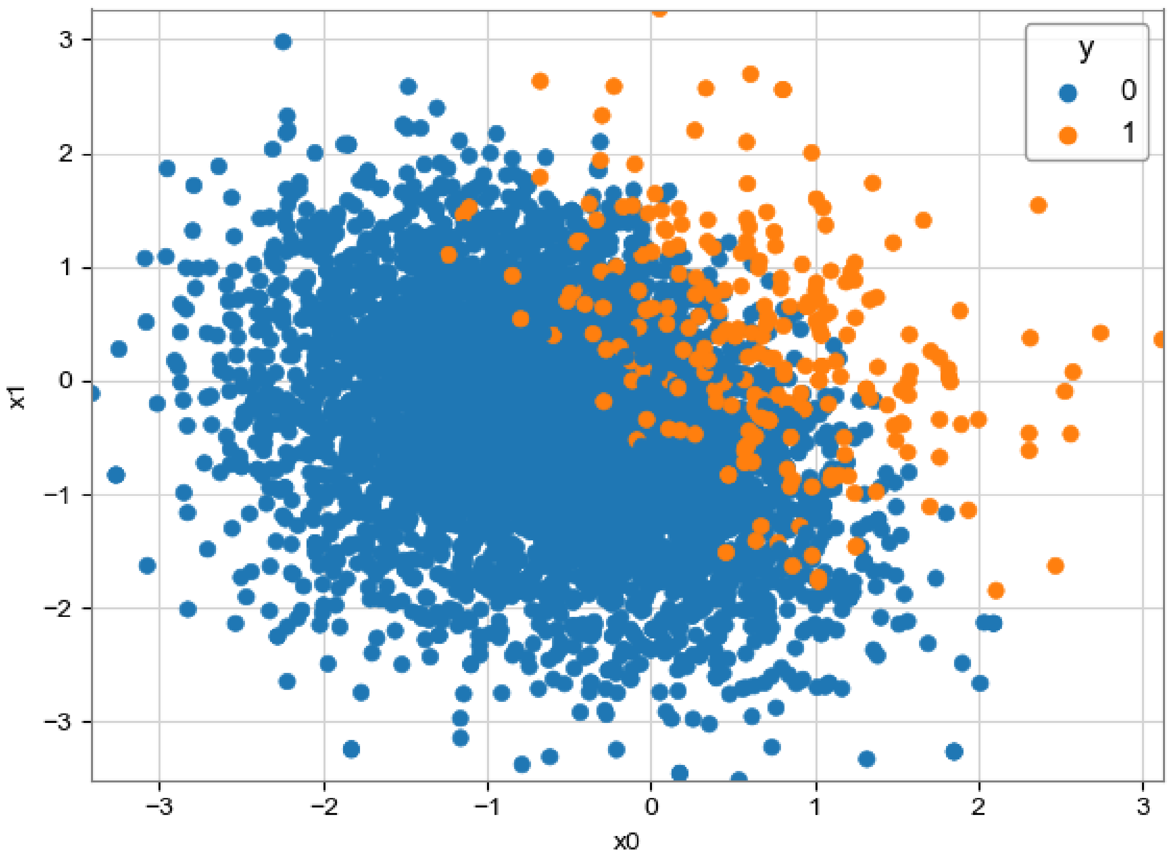
https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/
https://medium.com/james-blogs/handling-imbalanced-data-in-classification-problems-7de598c1059f
https://cluster-over-sampling.readthedocs.io/en/latest/auto_examples/plot_cluster_oversampler.html

Imbalanced dataset: Sampling

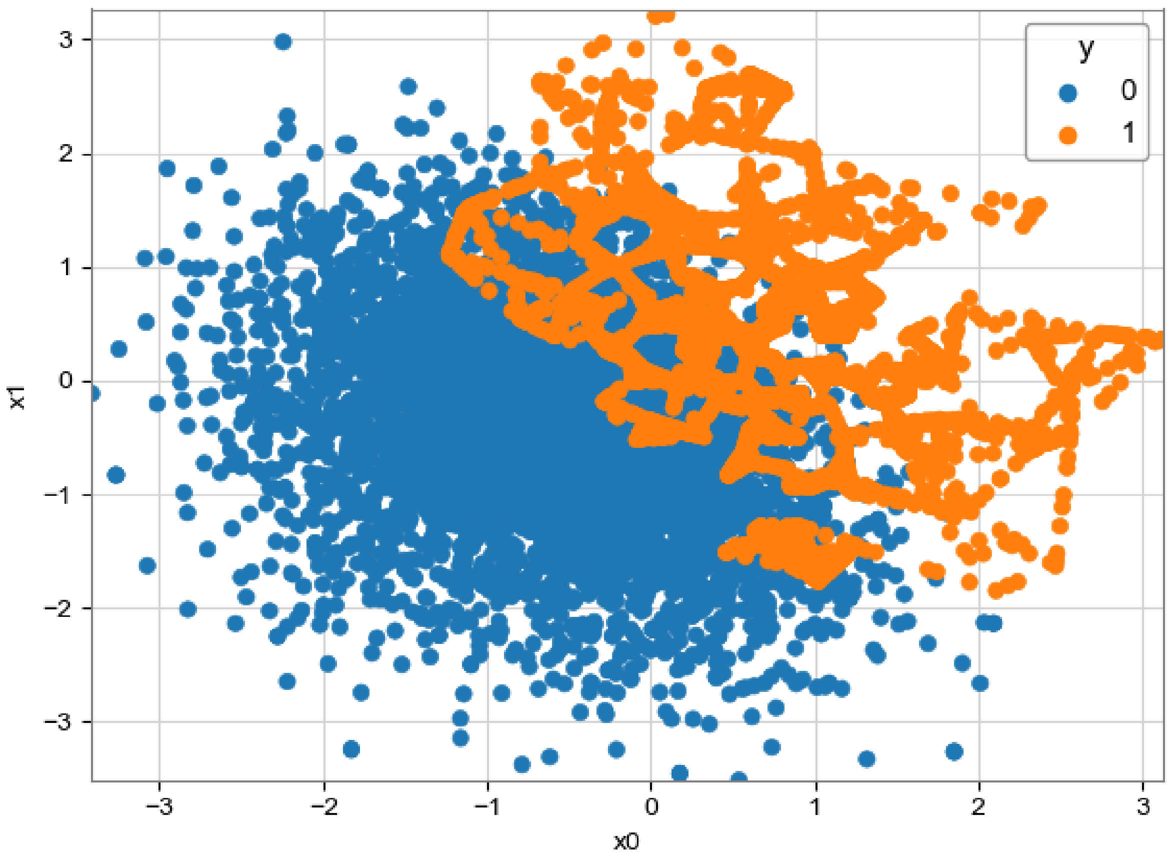
- **Cluster-Based Over Sampling** – In this case, the K-means clustering algorithm is independently applied to minority and majority class instances. This is to identify clusters in the dataset. Subsequently, each cluster is oversampled such that all clusters of the same class have an equal number of instances and all classes have the same size

- **Synthetic Minority Over-sampling Technique (SMOTE) –** A subset of data is taken from the minority class as an example and then new synthetic similar instances are created which are then added to the original dataset. This technique is good for Numerical data points.

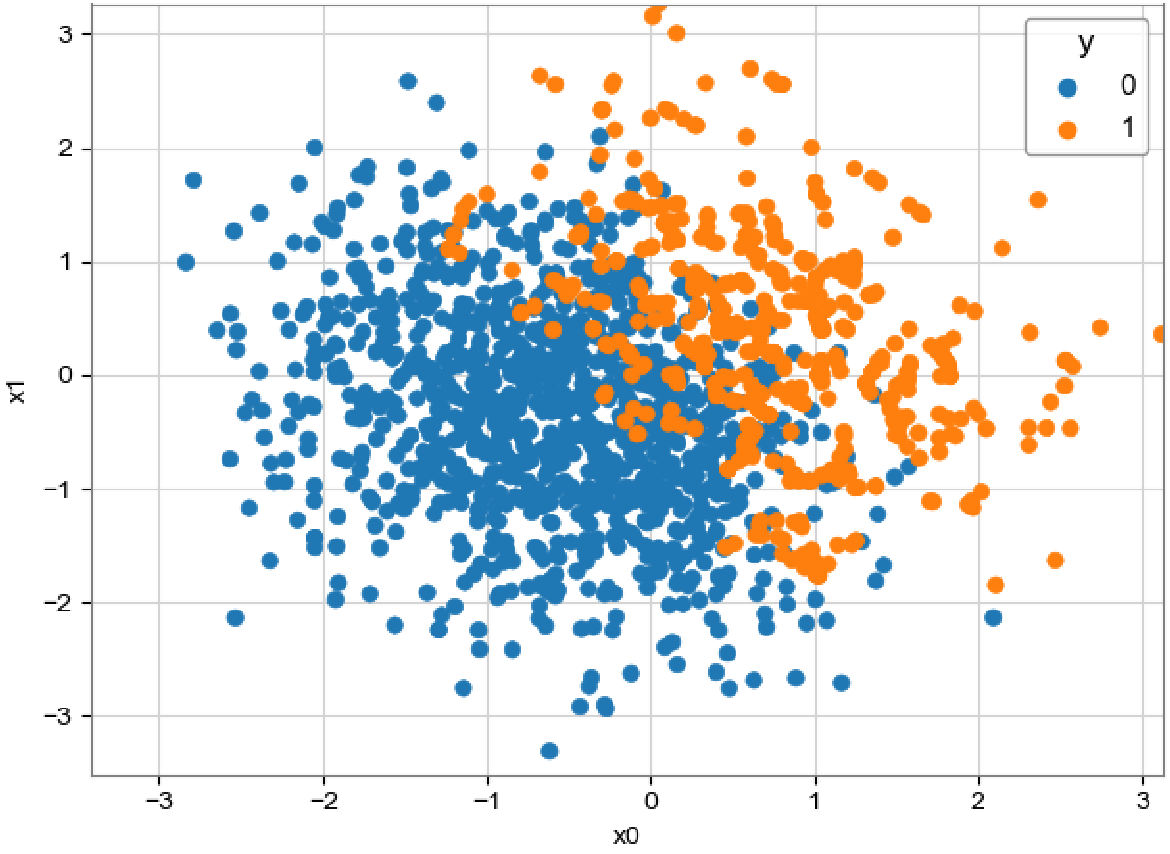
Imbalanced dataset: original



Imbalanced dataset: SMOTE

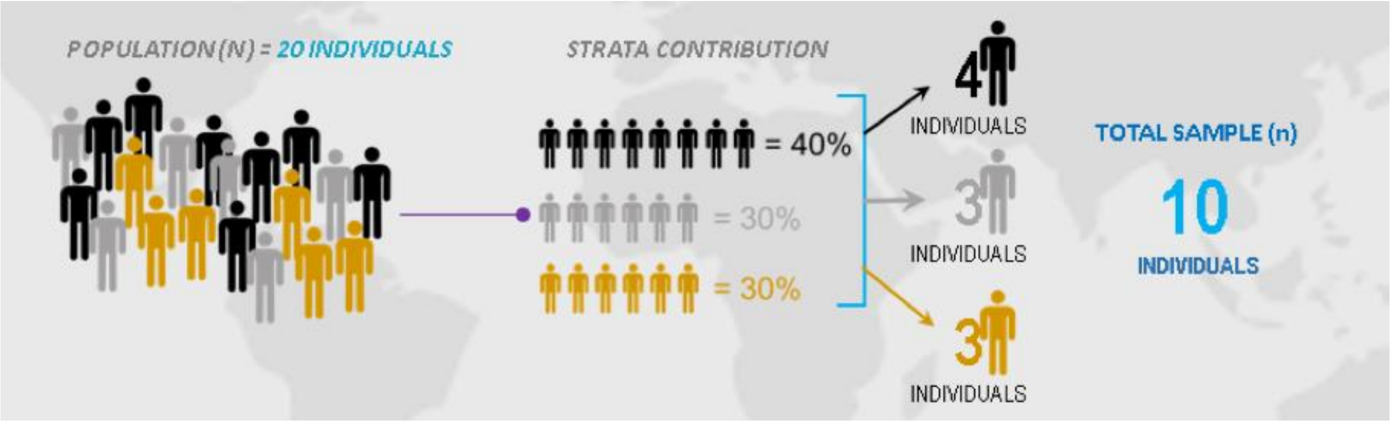


Imbalanced dataset: SMOTE + Undersample



Stratified Sampling

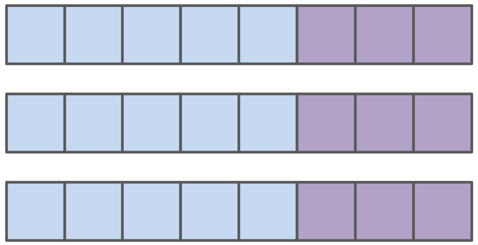
- In stratified sampling, researchers divide subjects into subgroups called strata based on characteristics that they share (e.g., race, gender, educational attainment, etc).
- Once divided, each subgroup is randomly sampled using another probability sampling method (so every member of the target population has a known chance of being included in the sample).



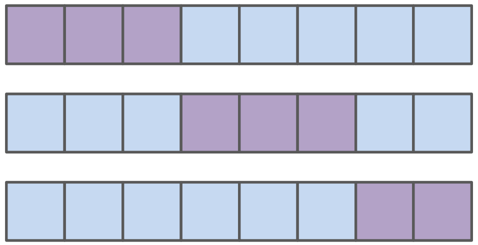
Cross Validation

Main types of cross validation techniques

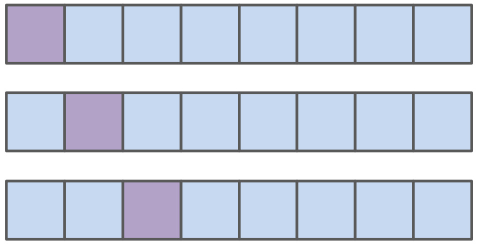
- K fold
- Stratified k fold
- Leave one out
- Bootstrapping



Holdout

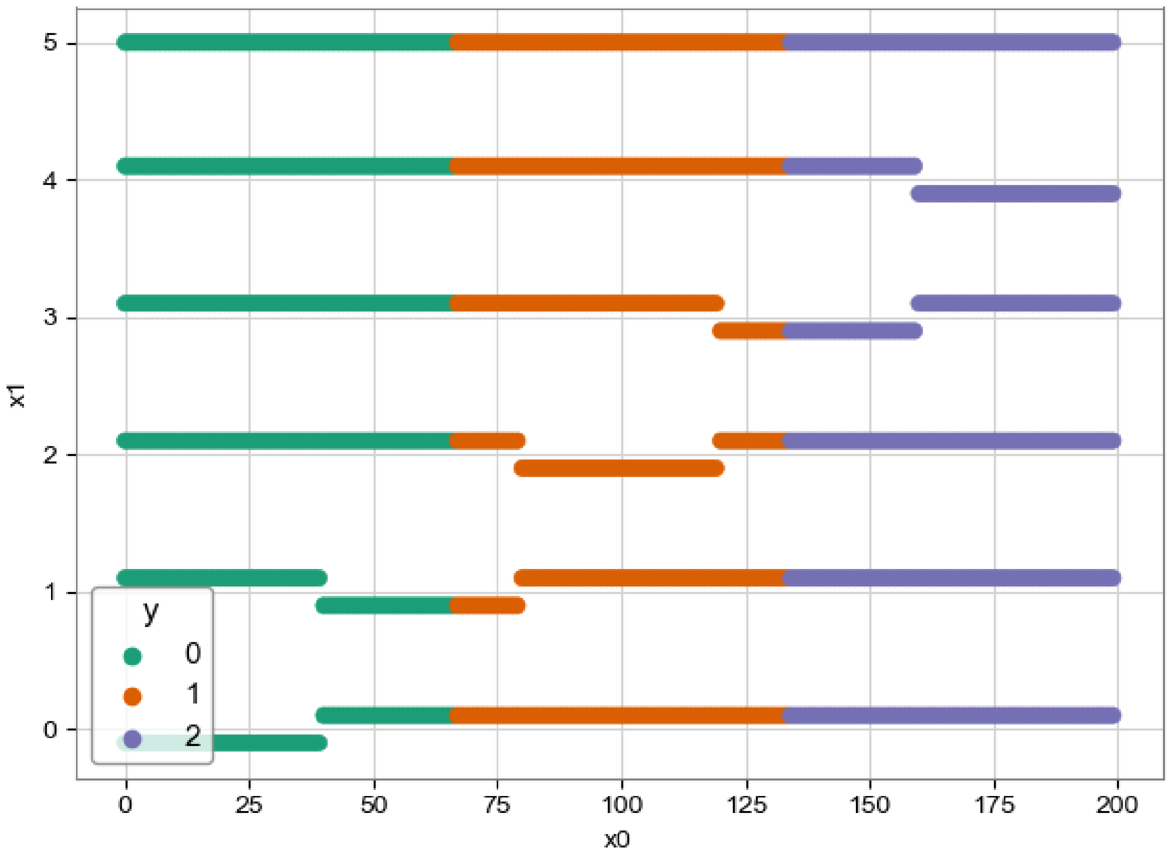


K-fold

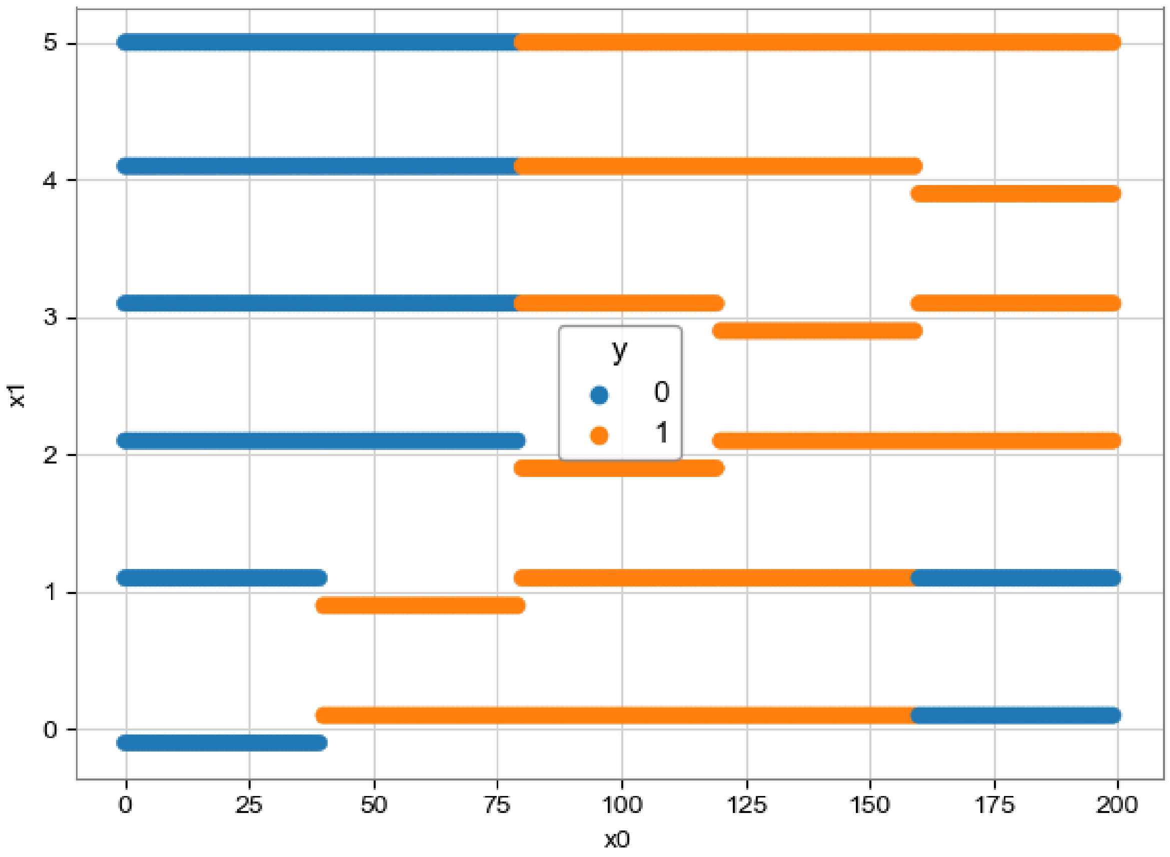


Leave one out

K-Fold



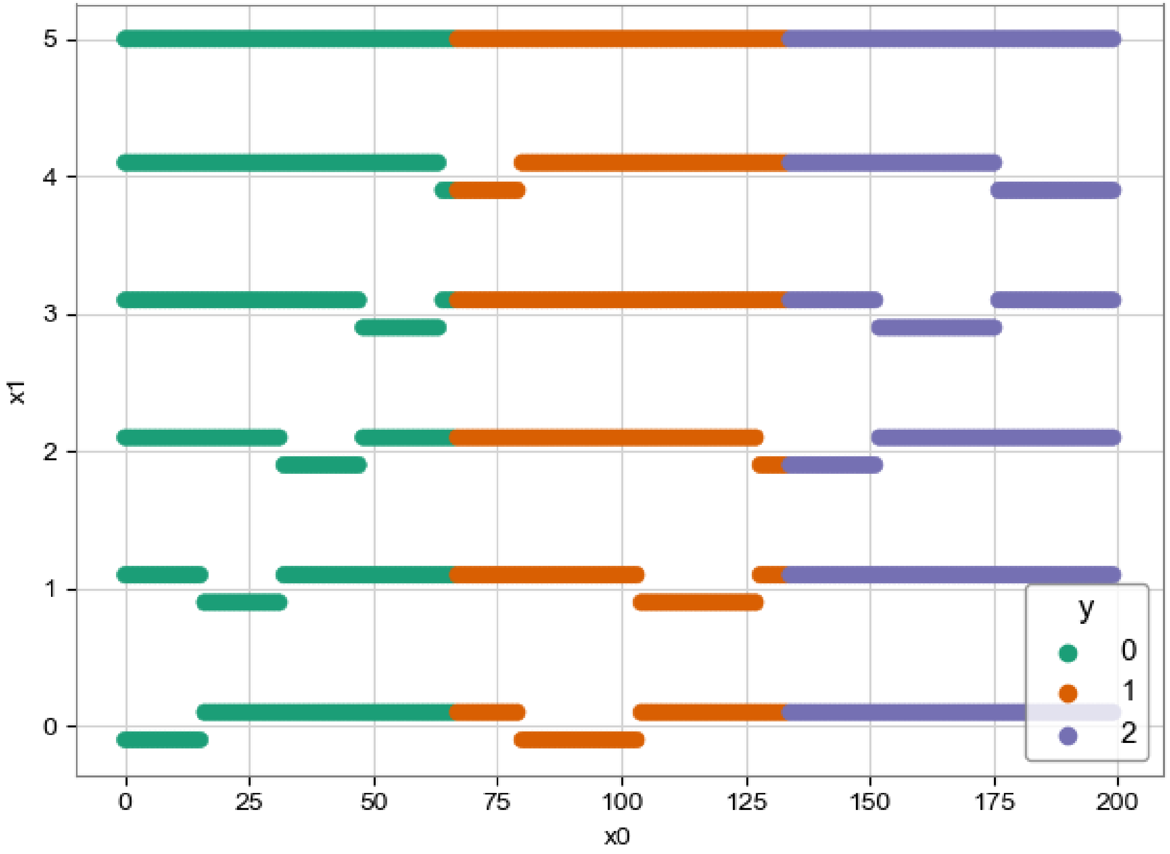
x1
x0
0
25
50
75
100
125
150
175
200
0
1
2
3
4
5
y
0
1
2

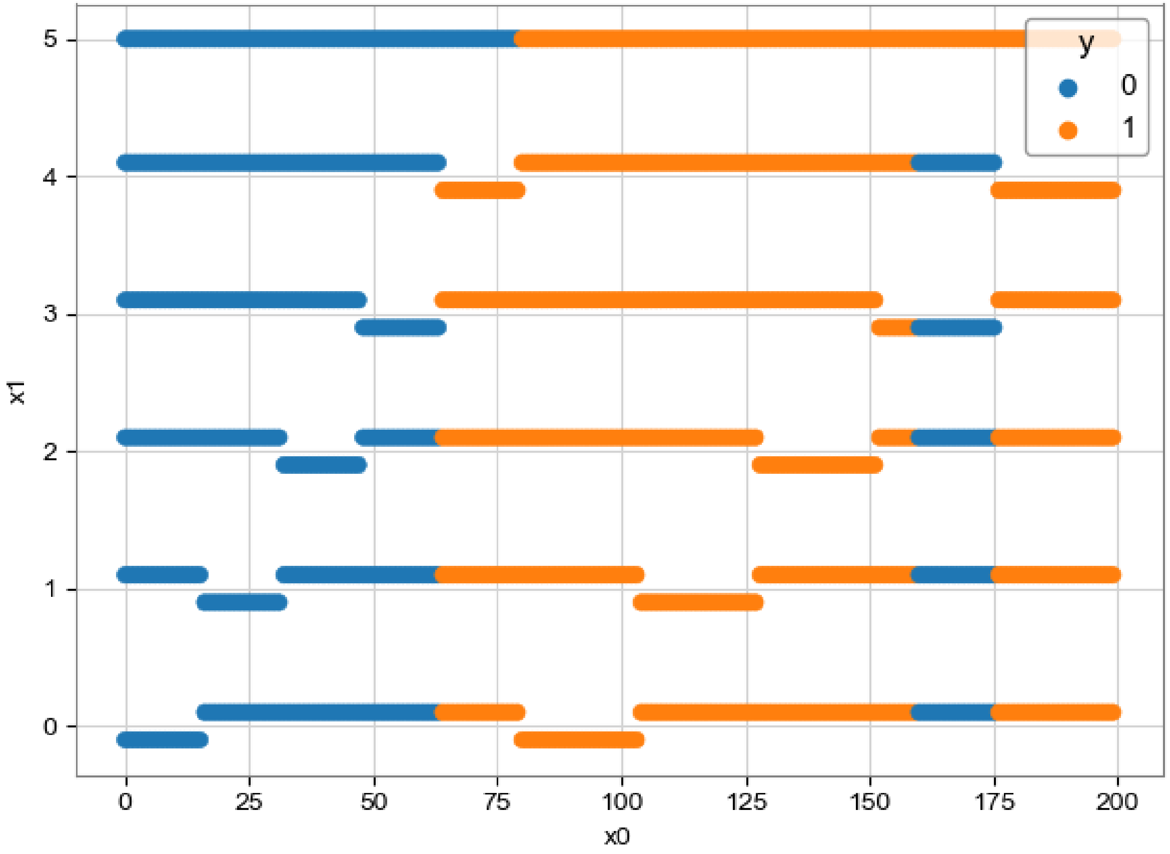


x1
x0
0
25
50
75
100
125
150
175
200
0
1
2
3
4
5
y
0
1

Cross Validation

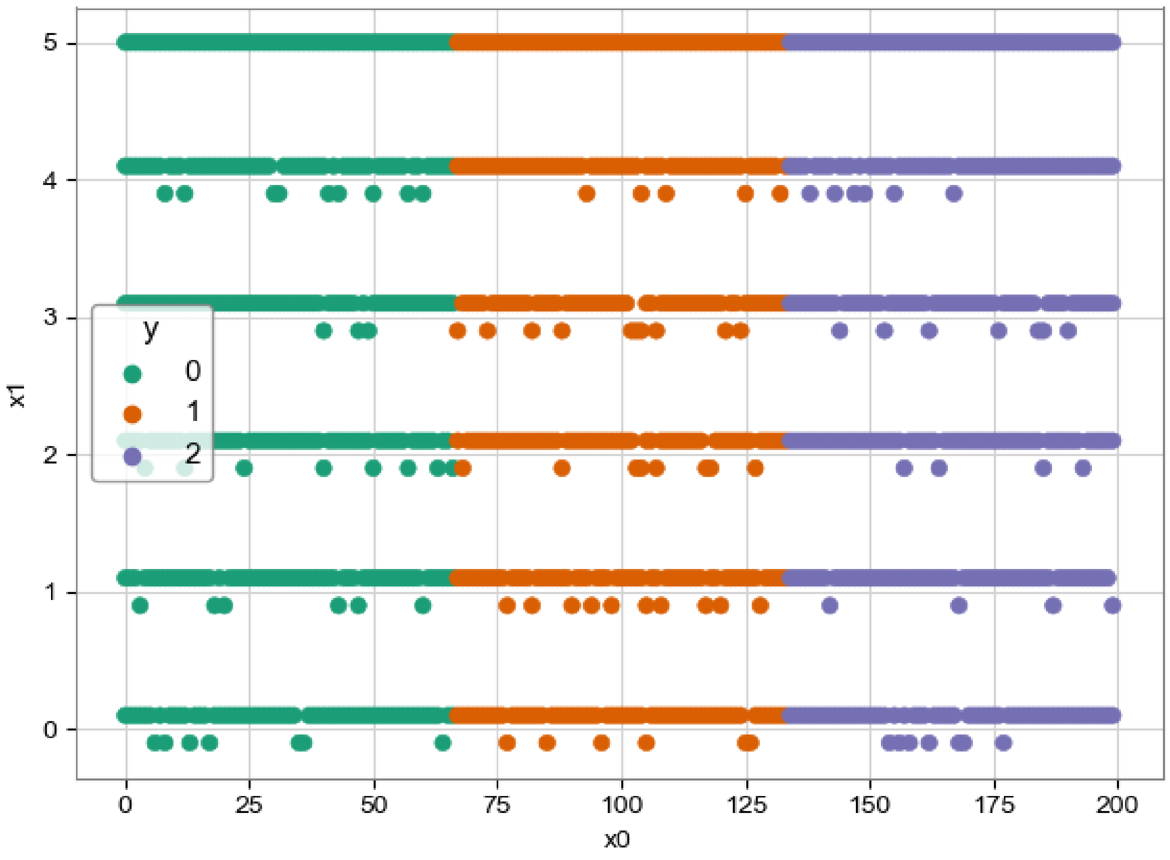
Stratified K-Fold

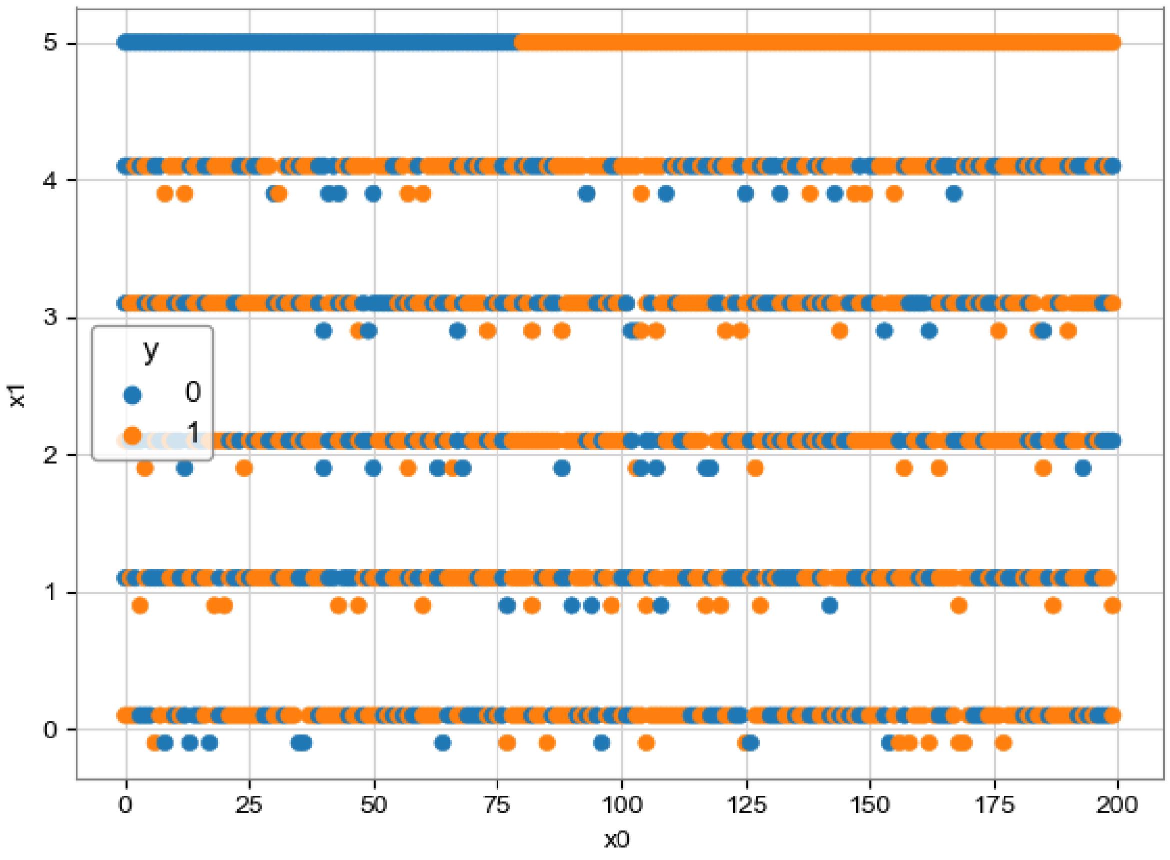




Cross Validation

Shuffle





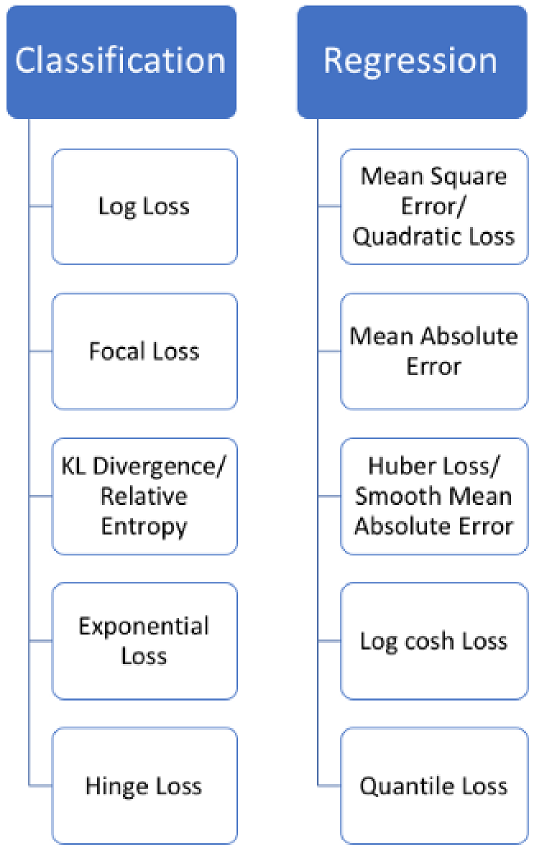
Learning curve

A learning curve, sometimes called a training curve, shows how the prediction score of training and validation sets depends on the number of training samples.

You can use learning_curve to get this dependency, which can help you find the optimal size of the training set, choose hyperparameters, compare models, and so on.

Loss functions

Types of loss



https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0

More

Normalization and Standardization

Normalization and Standardization are the two very popular methods used for feature scaling.

- Normalization refers to re-scaling the values to fit into a range of [0,1].
- Standardization refers to re-scaling data to have a mean of 0 and a standard deviation of 1 (Unit variance).
- Normalization is useful when all parameters need to have the identical positive scale however the outliers from the data set are lost. Hence, standardization is recommended for most applications.

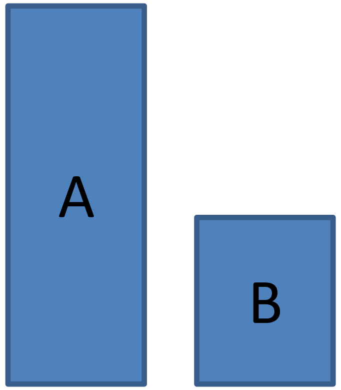
Distributions

The most popular distribution curves are as follows

- Bernoulli Distribution
- Uniform Distribution
- Binomial Distribution
- Normal Distribution
- Poisson Distribution
- Exponential distribution

Bernoulli Distribution

- Check if a team will win a championship or not
- A newborn child is either male or female
- You either pass an exam or not, etc.

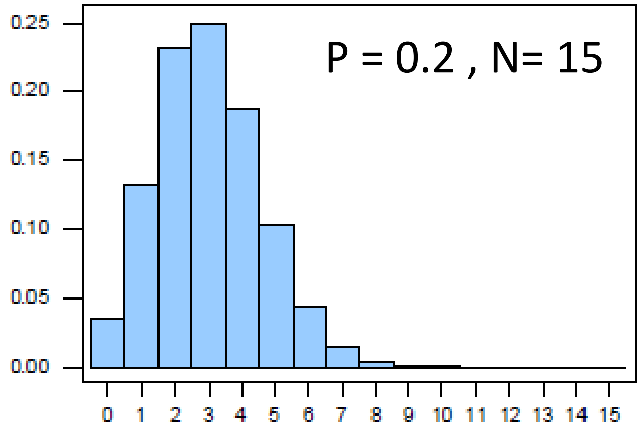


Uniform Distribution

- Fixed number of outcomes with uniform probability
- Rolling a single dice

Binomial Distribution

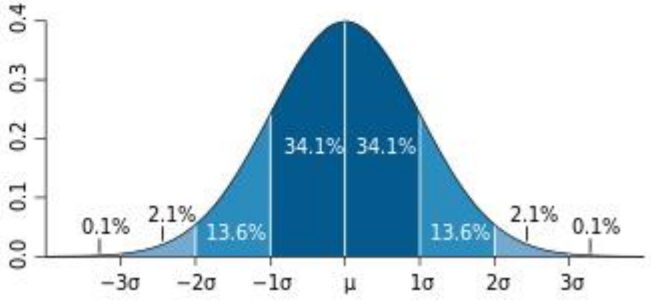
- Number heads in series of a coin toss



Feature Engineering

Normal Distribution

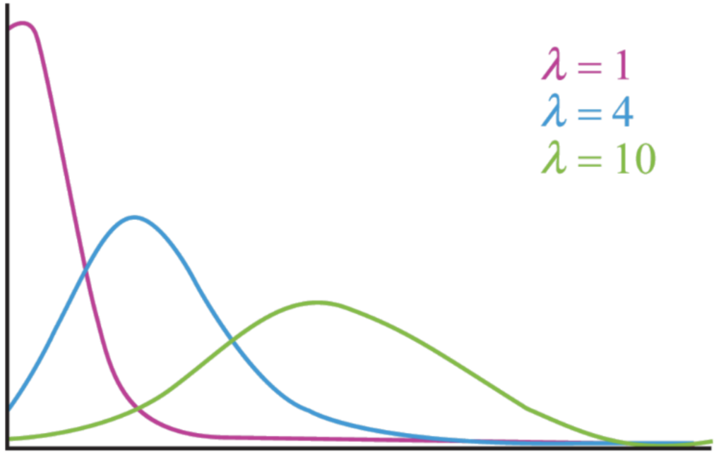
- Symmetric distribution where most of the observations cluster around the central peak.
- The values further away from the mean taper off equally in both directions.
- An example would be the height of students in a classroom.



Poisson Distribution

- Probability of a given number of events occurring in a fixed time/space interval
- events occur with a known constant mean rate
- events occur independently of the time since the last event.

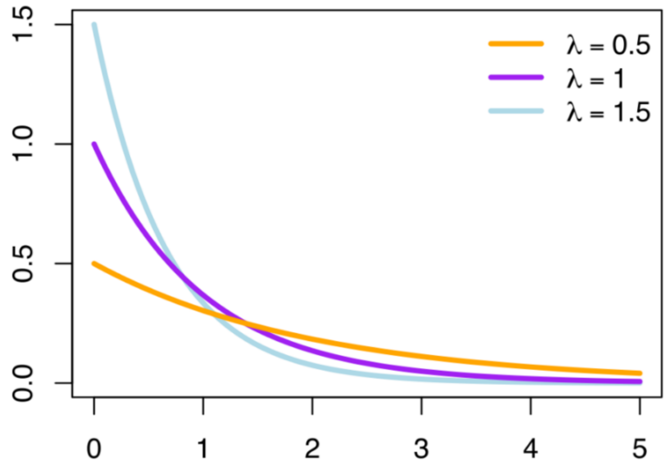
- It can be used to make e.g. forecasts about the number of customers on certain days and allows them to adjust supply according to the demand.



Exponential Distribution

- Concerned with the amount of time until a specific event occurs.

- For example, how long a car battery would last, in months



How to measure correlation of categorical data?

Chi square test can be used for doing so.
It gives the measure of correlation between categorical predictors.

https://datascience.stackexchange.com/questions/893/how-to-get-correlation-between-two-categorical-variable-and-a-categorical-variab