

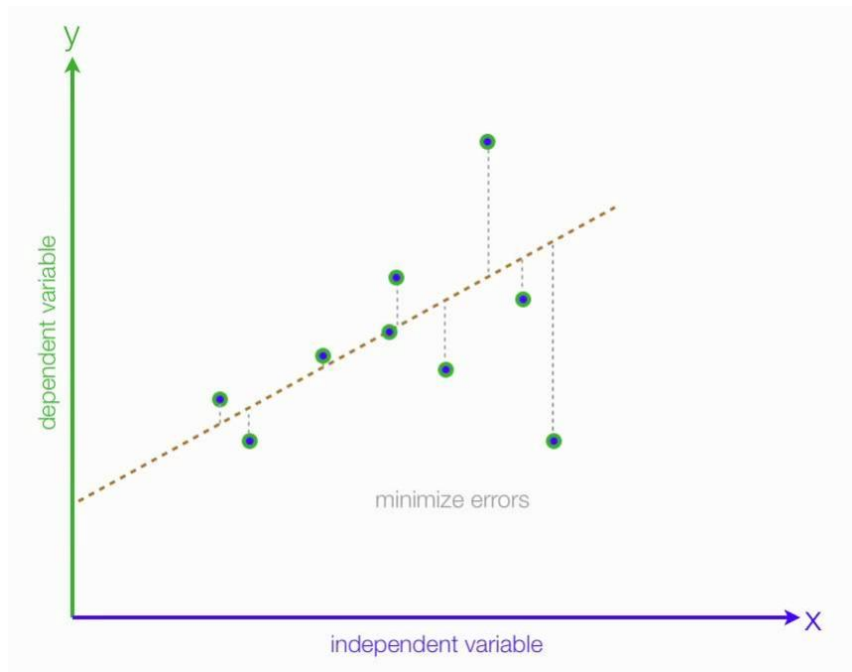
Enthuse Workshop

Intro to Data Science

Supervised Learning: Linear Regression

Linear regression

— — —



$$y = ax + b$$

Linear regression: preparing data

-- --

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

Predicting the profit using linear regression

— — —

We need to split the dataset into the training set and test set:

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,  
random_state = 0)
```

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

Predicting the profit using linear regression

— — —

And then fit a regression model:

```
from sklearn.linear_model import LinearRegression  
  
regressor = LinearRegression()  
  
regressor.fit(X_train, y_train)
```

Predicting the profit using linear regression

— — —

Test the model:

```
y_pred = regressor.predict(X_test)
```

Transform discrete variables

Linear regression: preparing data

— — —

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

- the 4th column is text.
- we can't have text in our data if we're going to run any kind of model on it.
- we need to make this data ready for the model: convert it to numbers

preparing data: label encoder

— — —

Each distinct name should be replaced by a number

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94



0	165349.20	136897.80	471784.10	2.0
1	162597.70	151377.59	443898.53	0.0
2	153441.51	101145.55	407934.54	1.0
3	144372.41	118671.85	383199.62	2.0
4	142107.34	91391.77	366168.42	1.0

preparing data: label encoder

— — —

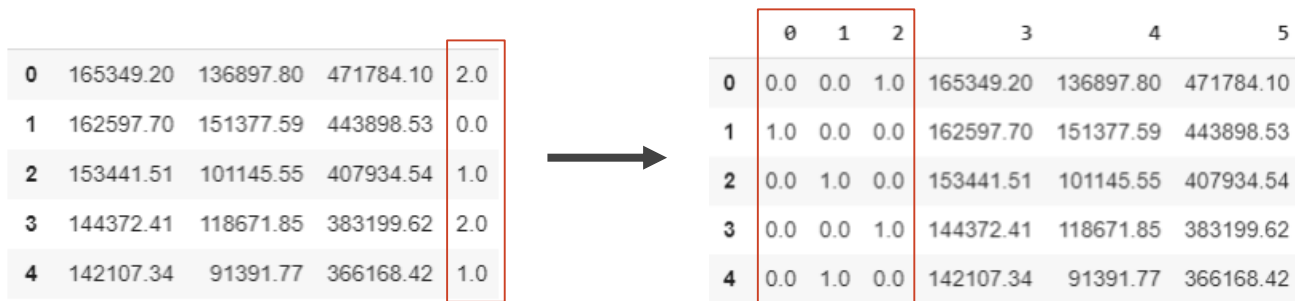
Code:

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder  
  
labelencoder = LabelEncoder()  
  
X[:, 3] = labelencoder.fit_transform(X[:, 3])
```

Problem with our encoding


— — —

- The data is categorical and does not have any relation between different categories
- However, our encoding implies a relation: $0 < 1 < 2$.
- To overcome this problem, we use One Hot Encoder:



The diagram illustrates the process of converting a categorical variable into a one-hot encoded matrix. On the left, a table shows five data points with a categorical variable represented by values 0, 1, 2, 3, and 4. A red box highlights the column for this variable. An arrow points to the right, where the same data is shown, but the categorical variable is replaced by a one-hot encoded matrix. This matrix has five columns, each corresponding to a category (0, 1, 2, 3, 4). Each row contains a 1 in the column corresponding to its category and 0s elsewhere. A red box highlights the first three columns of this matrix.

0	165349.20	136897.80	471784.10	2.0
1	162597.70	151377.59	443898.53	0.0
2	153441.51	101145.55	407934.54	1.0
3	144372.41	118671.85	383199.62	2.0
4	142107.34	91391.77	366168.42	1.0



	0	1	2	3	4	5
0	0.0	0.0	1.0	165349.20	136897.80	471784.10
1	1.0	0.0	0.0	162597.70	151377.59	443898.53
2	0.0	1.0	0.0	153441.51	101145.55	407934.54
3	0.0	0.0	1.0	144372.41	118671.85	383199.62
4	0.0	1.0	0.0	142107.34	91391.77	366168.42

One-hot encoding

— — —

Code:

```
onehotencoder = OneHotEncoder(categorical_features = [3])
```

```
X = onehotencoder.fit_transform(X).toarray()
```

Thank you!