Food Sales Prediction

Goran Topic

In this presentation we take from a food store and we use data science to make a prediction about the sales.

Coding Dojo

1. Understand our data
2. Clean our data
3. Visualize our data
4. Find a Model
5. Make a Prediction

1

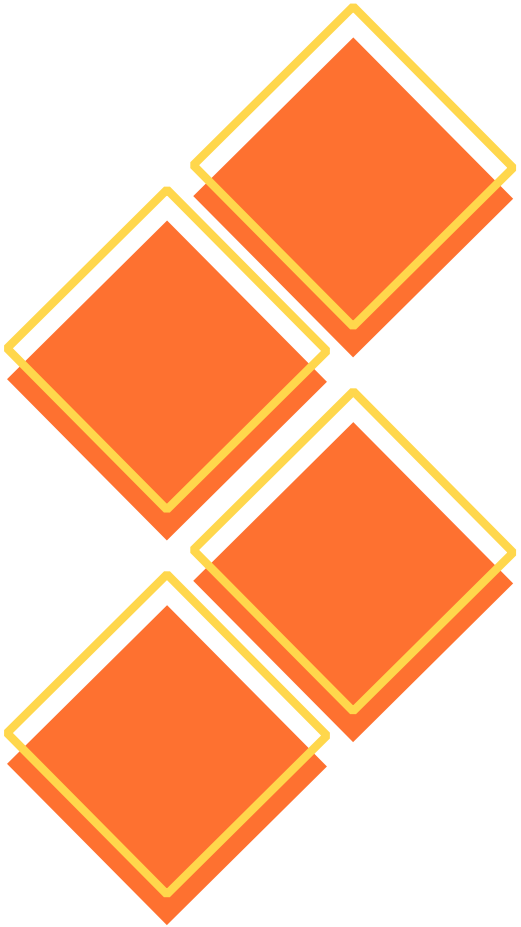
UNDERSTANDING OUR DATA

First let’s try to understand what kind of data do we have.
Here is a the first rows of our data

Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier
FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049
DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018
FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049
FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010
NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013

First let’s try to understand what kind of data do we have.

Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380
OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228
OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700
OUT010	1998	NaN	Tier 3	Grocery Store	732.3800
OUT013	1987	High	Tier 3	Supermarket Type1	994.7052



What kind of Data type are the Columns?

```
sales_df.dtypes
```

0s

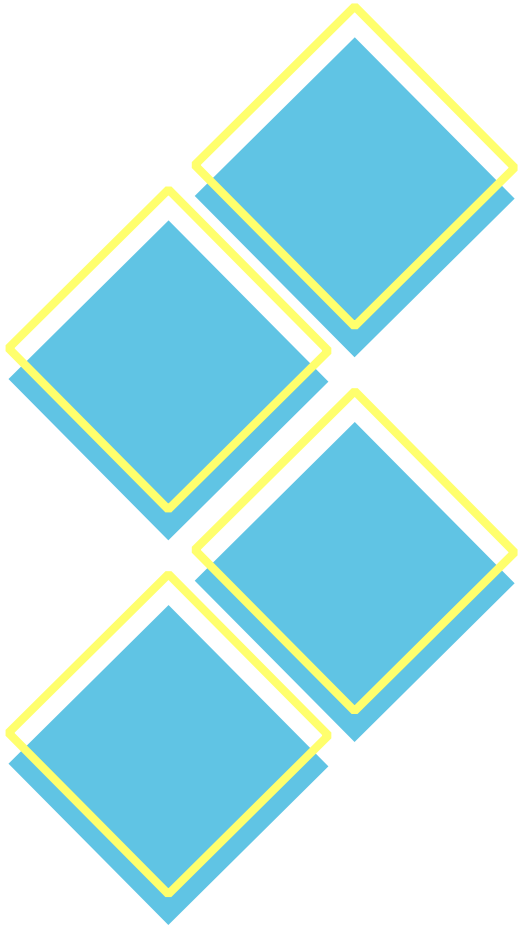
```
Item_Identifier              object
Item_Weight                 float64
Item_Fat_Content             object
Item_Visibility             float64
Item_Type                    object
Item_MRP                    float64
Outlet_Identifier            object
Outlet_Establishment_Year     int64
Outlet_Size                  object
Outlet_Location_Type         object
Outlet_Type                  object
Item_Outlet_Sales           float64
dtype: object
```

2 Cleaning Data

Let’s check which columns have missing values.

```
# lets fin the number of colmns which have the missing values
sales_df.isna().any()
```

```
Item_Identifier              False
Item_Weight                   True
Item_Fat_Content             False
Item_Visibility              False
Item_Type                    False
Item_MRP                     False
Outlet_Identifier            False
Outlet_Establishment_Year    False
Outlet_Size                   True
Outlet_Location_Type         False
Outlet_Type                  False
Item_Outlet_Sales            False
dtype: bool
```



We can see that the Columns which have missing values is just `Item_Weight` and `Outlet_Size`

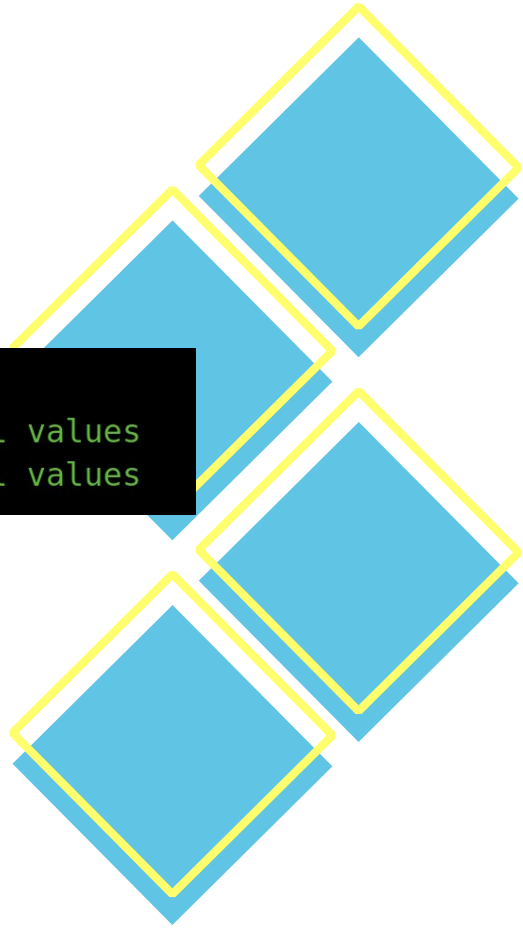
2 Dropping Columns

What percentage of our data is missing?

```
# percentage of NA values
print(2410 / 8525 * 100 ) # this is about %28 percent of all values
print(1463 / 8525 * 100 ) # this is about %17 percent of all values
```

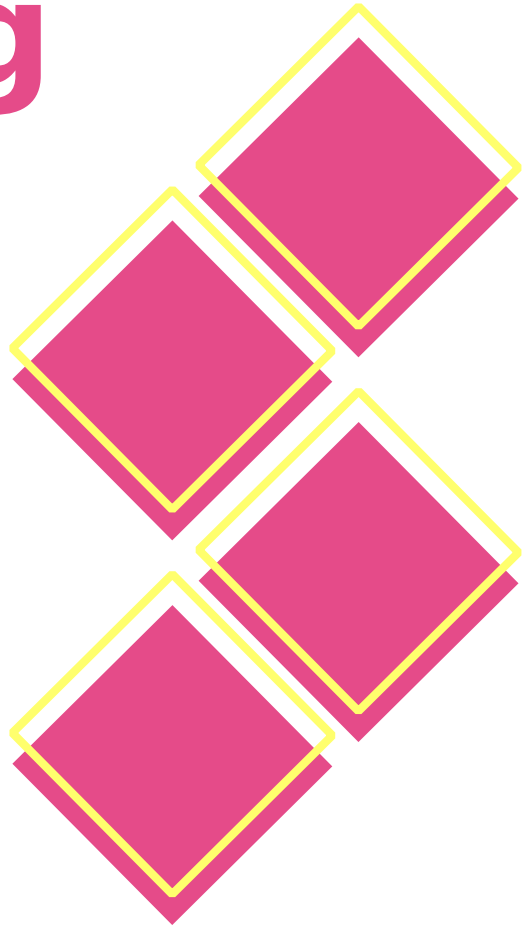
```
clean_df = sales_df.dropna()
# check if there is nay na value
clean_df.isna().any().any()
```

```
False
```



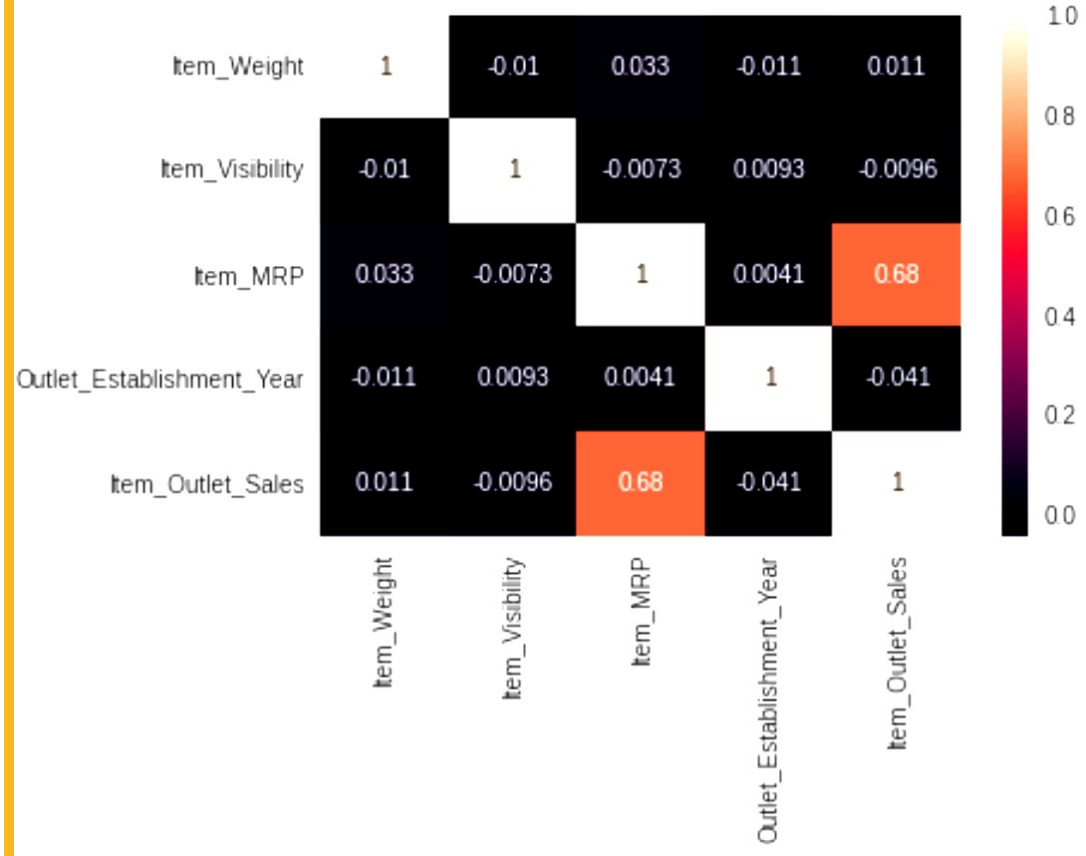
3

Visualizing the Data

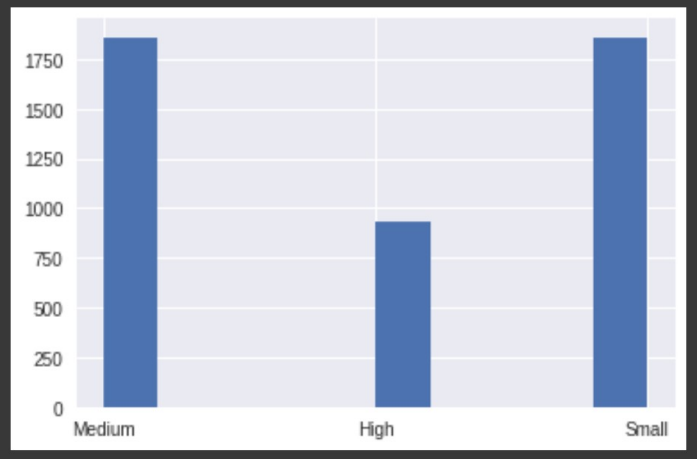


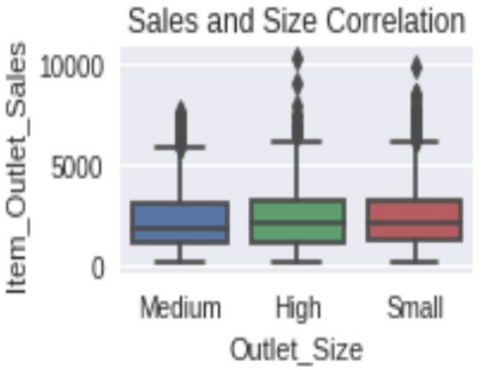
Let's try to understand our data a bit more by visualizing it

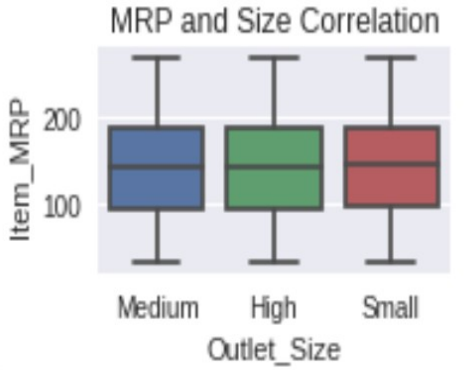

ICE CREAM

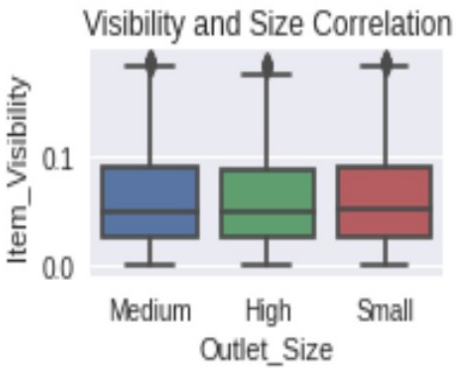

Item_Weight
Item_Visibility
Item_MRP
Outlet_Establishment_Year
Item_Outlet_Sales
1
-0.01
0.033
-0.011
0.011
-0.01
1
-0.0073
0.0093
-0.0096
0.033
-0.0073
1
0.0041
0.68
-0.011
0.0093
0.0041
1
-0.041
0.011
-0.0096
0.68
-0.041
1
1.0
0.8
0.6
0.4
0.2
0.0

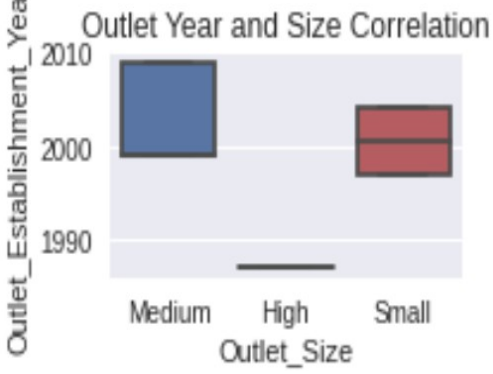
```
clean_df.Outlet_Size.hist();
```

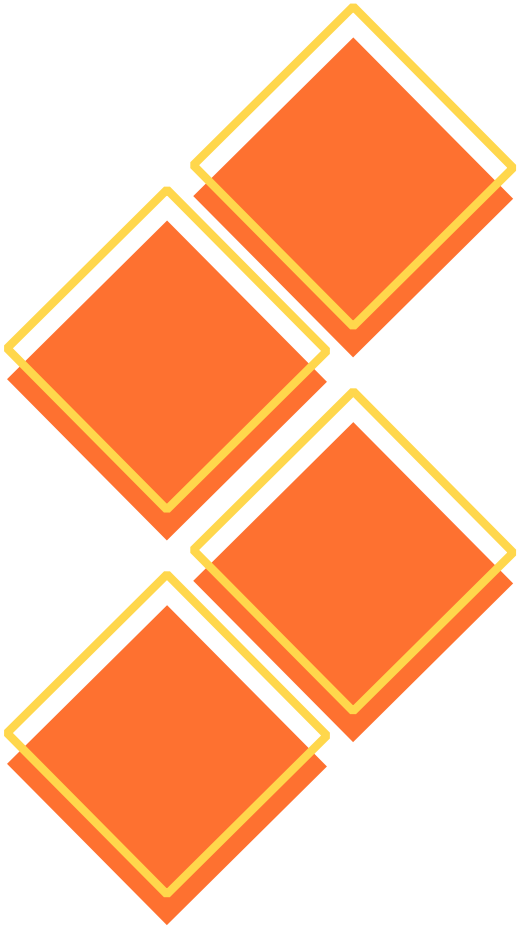






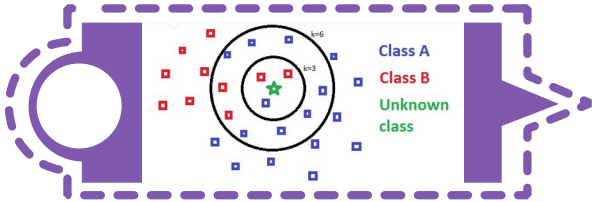






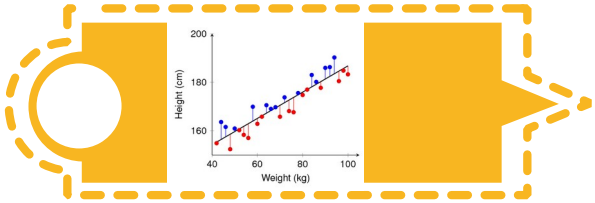
4

Finding a Model



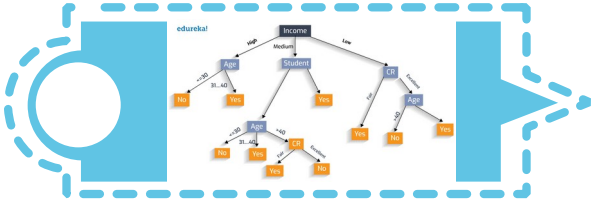
Finding Nearest Neighbors

At the closes data for help.



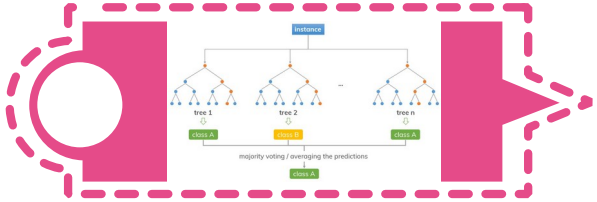
Regression

A line which has the lease amount to error margin.



Decision Tree

A Tree which uses regression to fork different branches of the data.



Random Forest

A group of Decision Trees randomly generated.

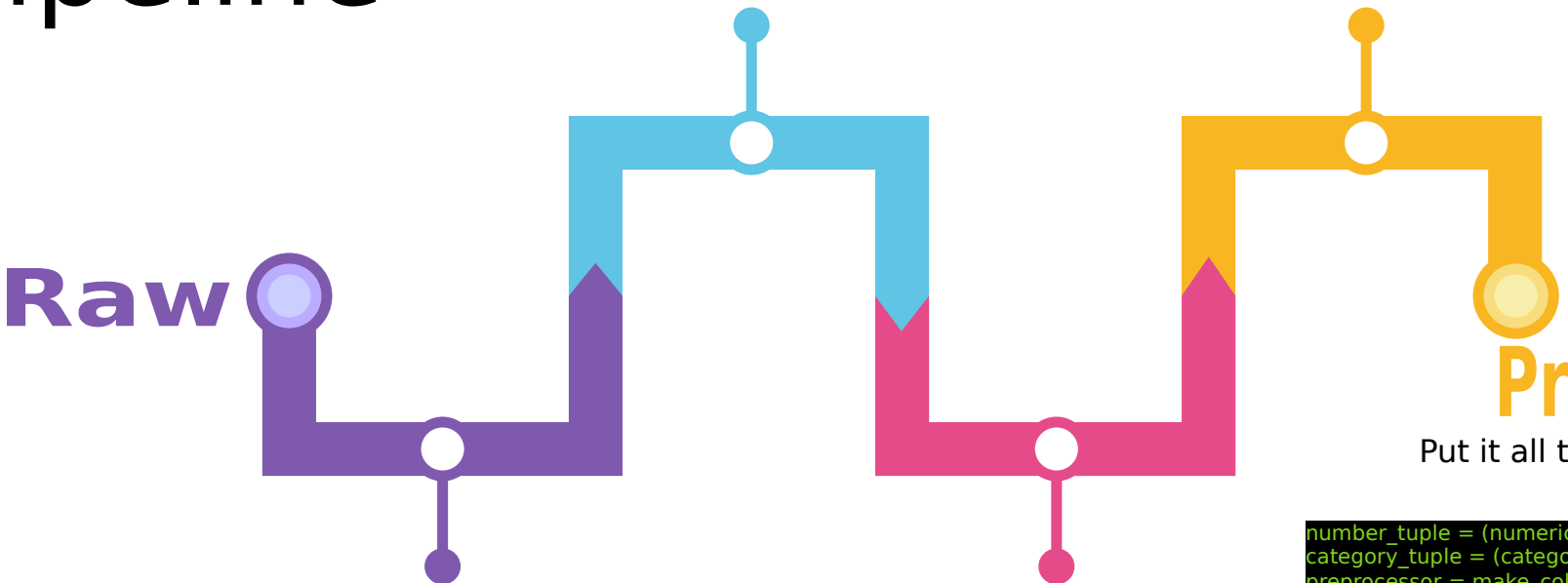
The Pipeline

```
cat_selector =
make_column_selector(dtype_include='object')
num_selector =
make_column_selector(dtype_include='number')
```

Make numeric an categorical selectors

```
ohe = OneHotEncoder(handle_unknown='ignore',
sparse=False)                                    scaler =
StandardScaler()
```

Make One hot encoding and a Scaler



Split the data into training and testing

```
X_train, X_test, y_train, y_test = train_test_split(features,
target, random_state=42
```

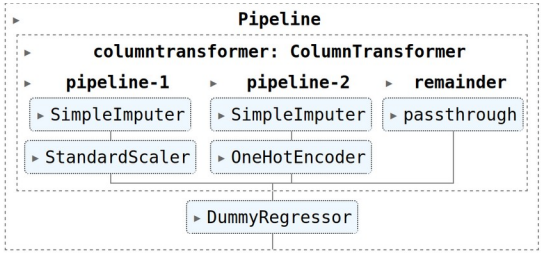
Create the imputers to handle missing data

```
freq_imputer = SimpleImputer(strategy='most_frequent')
mean_imputer = SimpleImputer(strategy='mean')
```

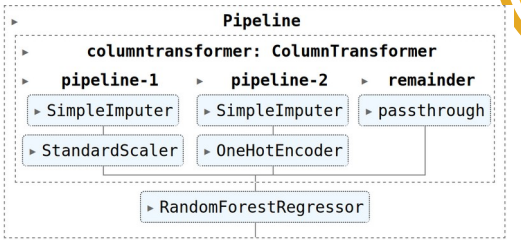
Put it all together

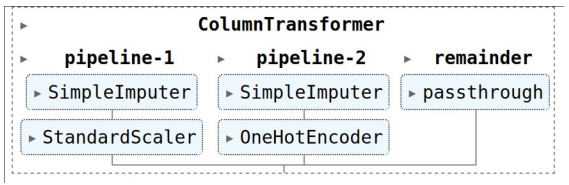
```
number_tuple = (numeric_pipe, num_selector)
category_tuple = (categorical_pipe, cat_selector)
preprocessor = make_column_transformer(number_tuple,
category_tuple)
```

Dummy Regressor

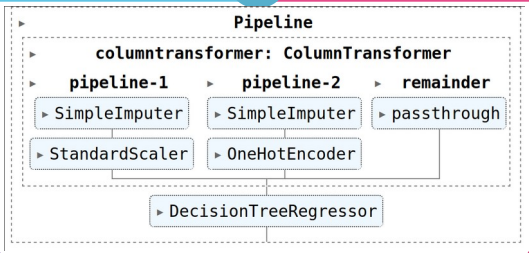


Random Forest





Normal Pipeline



Decision Tree

Finding the best prediction

```
# maybe this was a bad idea
train_r2 = []
test_r2 = []
for depth in range(1, 100):
  print(depth)
  forest = RandomForestRegressor(
      max_depth = depth,
      random_state = 42
      )
  pipe = make_pipeline(
      preprocessor, forest
      )
  pipe.fit(X_train, y_train)
  train_r2.append(
      pipe.score(X_train, y_train) )
  test_r2.append(
      pipe.score(X_test, y_test) )

f'the optimal depth is: {test_r2.index(min(test_r2))}\
 with a test score of {min(test_r2)}'
```


THANK YOU