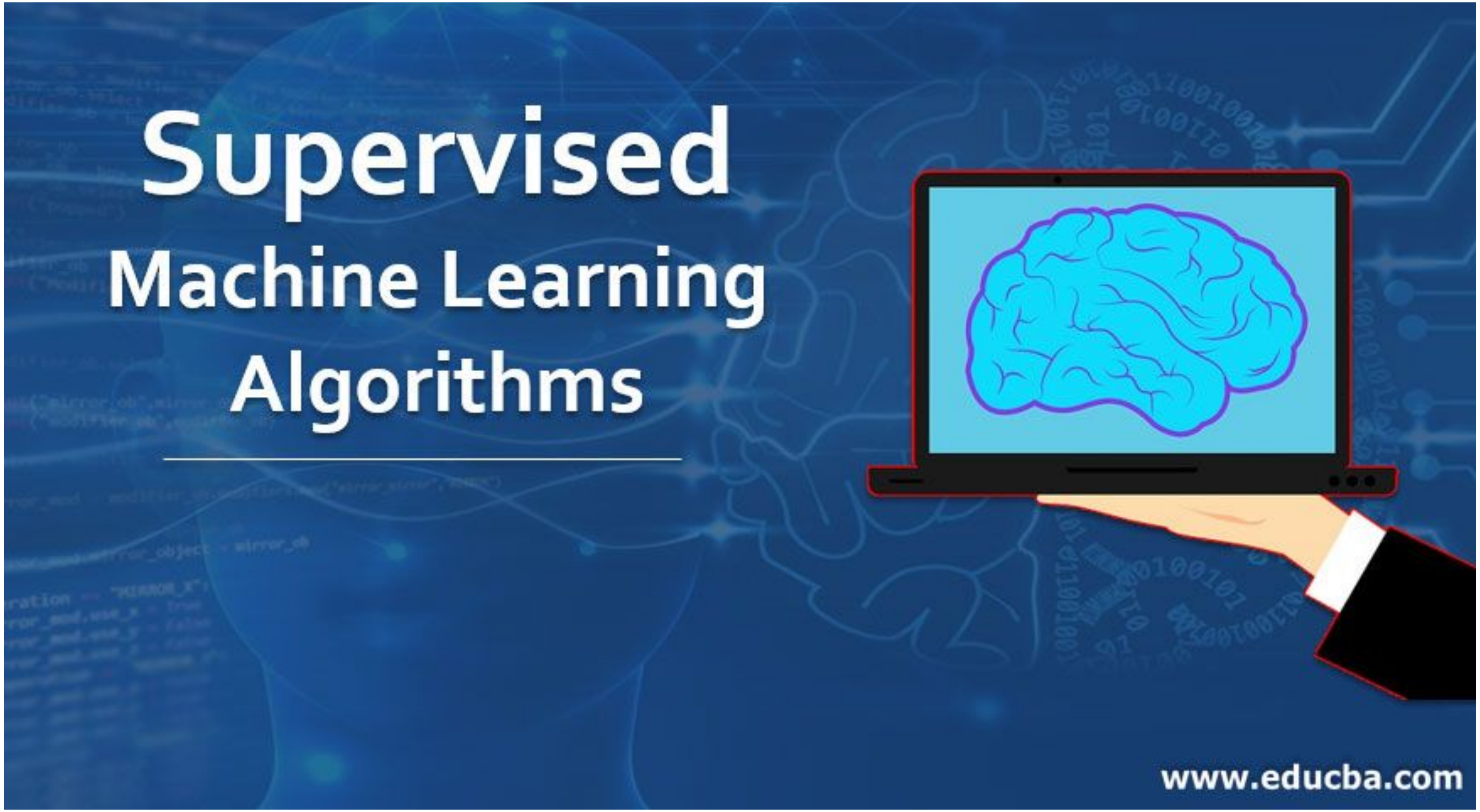


ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

Last lecture reminder

We learned about:

- Solving simple linear regression - Theory & practical
- Solving multiple linear regression - Theory
- Solving simple linear regression using Python
- Polynomial regression

SciPy

Introduction To SciPy

SciPy (Scientific Python) is a free and open-source Python library used for high-level technical and scientific computing. It is built on the NumPy extension and allows the user to manipulate and visualize data with a wide range of high-level commands.

Main features in the SciPy library:

- SciPy builds on NumPy and provides a large set of scientific and mathematical functions that operate on numpy arrays which makes it easy to use and compute.
- SciPy contains modules for linear algebra, interpolation, integration, optimization, statistics, special functions, signal and image processing, etc
- SciPy provides fast N-dimensional array manipulation. The key functionality provided by SciPy library is the ability to manipulate and visualize mathematical data.
- SciPy is built upon the Python eco-system and Python being one of the most used programming languages for Data Science, SciPy becomes an optimal choice for data scientists.

scikit learn

python

Introduction To Scikit-Learn

Scikit-Learn is an open-source machine learning library for Python. It features various classification, regression, and clustering algorithms, including support vector machines, random forests and k-means. It is designed to interoperate with the Python scientific and numerical libraries NumPy and SciPy.

Main features in the Scikit-Learn library:

- Scikit-Learn provide simple and efficient tools for predictive data analysis.
- It’a generalized “estimator API” framework allow a simple way to call and execute different machine learning models.
- The library comes with many convenience tools, including train test split functions, cross validation tools and a variety of reporting metric functions.
- Scikit-Learn library is built on NumPy, SciPy, and matplotlib.

Packages Installation And Import

In order to install Scikit-Learn and SciPy package we should run the following commands:

```
pip install scipy

pip install -U scikit-learn
```

In order the validate the the installation finished successfully, you can run on your Jupyter notebook:

```
import sklearn

import scipy
```

In case there are no errors, that means the the installation ended successfully and you can now use those packages in your supervised learning process.

Supervised Machine Learning Framework

To better understand the supervised machine learning process let's look at our previous example with predicting the house prices according to given features (area, number of bedrooms, number of Bathrooms).

In this example we had the historical data about each house features and the price it was sold for:

X			y
Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$450,000
230	3	3	$650,000
180	1	1	$400,000
210	2	2	$550,000

Supervised Machine Learning Framework

Now, the way we want to train our supervised learning model is by applying a **train / test split**.

Meaning we will divide our historical data to 2 section:

- **Train section** → This is the historical data that we will provide our machine learning algorithm and according to this data the algorithm will provide the predicted result.
- **Test section** → This is the historical data that we will provide our machine learning algorithm after the training is over in order to test the algorithm result with actual data.

	Area m^2	Bedrooms	Bathrooms	Price	
	200	3	2	$500,000	
X TRAIN	190	2	1	$450,000	Y TRAIN
	230	3	3	$650,000	
X TEST	180	1	1	$400,000	Y TEST
	210	2	2	$550,000	

Supervised Machine Learning Framework

When working with Scikit-Learn and appling supervised machine learning it’s important to work according to the following framework:

1. Import from Scikit-Learn the supervised learning model you want to use.
2. Create an instance of that model and provide the chosen parameters.
3. Train the model according to the x_train and y_train sets from your historical data.
4. Take the algorithm prediction to your x_test set from your historical data.
5. Import from Scikit-Learn the error metric you want to evaluate your model accuracy.
6. Compare your model predictions results to the actual y_test historical data and see the error metric value (there will always be an error because we never will get 100% correct predictions).
7. In case your error metric result is in the valid range you know that your model is ready for real unknown data. In case that your error metric result is not valid you will need to improve your algorithm accuracy and test it again.

Supervised Learning Framework - Python Example

Now that we know the basic framework for applying supervised machine learning let's take a real example to demonstrate how it's actually works.

For this example we'll use the **'Advertising.csv'** file that provide historical data on advertising campaign expenses and the sales result for each campaign.

```
In [4]: df = pd.read_csv('/Users/ben.meir/Downloads/UNZIP_FOR_NOTEBOOKS_FINAL/DATA/Advertising.csv')
        df

Out[4]:
```

	TV	radio	newspaper	sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	9.7
197	177.0	9.3	6.4	12.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	13.4

200 rows × 4 columns

In previous lecture we created a simple linear regression model combining all expenses to 'total_expenses'.

<u>This time we want to perform multiple linear regression with all features (TV, radio, newspaper).</u>

Supervised Learning Framework - Python Example

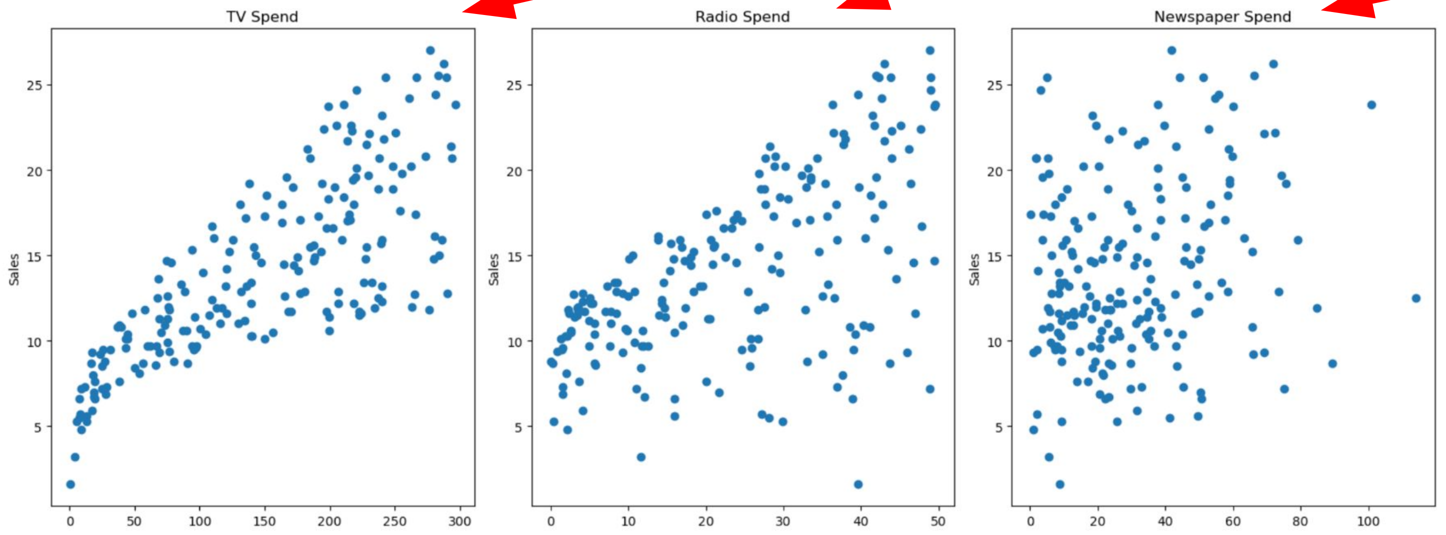
First, because we are now dealing with multiple features let's create a scatter plot for each feature and see its correlation to the campaign sales results.

```
In [10]: fig,axes = plt.subplots(nrows=1,ncols=3,figsize=(16,6))

         axes[0].plot(df['TV'],df['sales'],'o')
         axes[0].set_ylabel("Sales")
         axes[0].set_title("TV Spend")

         axes[1].plot(df['radio'],df['sales'],'o')
         axes[1].set_title("Radio Spend")
         axes[1].set_ylabel("Sales")

         axes[2].plot(df['newspaper'],df['sales'],'o')
         axes[2].set_title("Newspaper Spend");
         axes[2].set_ylabel("Sales")
         plt.tight_layout();
```



Tv and Radio spend has a clear linear correlation to the sales results

Newspaper spend doesn't have a clear correlation to the sales results

Supervised Learning Framework - Python Example

We can also examine relationships between features, for example the correlation between TV expenses and Radio expenses.

sns.pairplot() → The Seaborn pairplot() method allow us to generate a grid providing all combinations between different columns in the dataframe. Each plot will be a scatter plot representing the correlation between the columns.

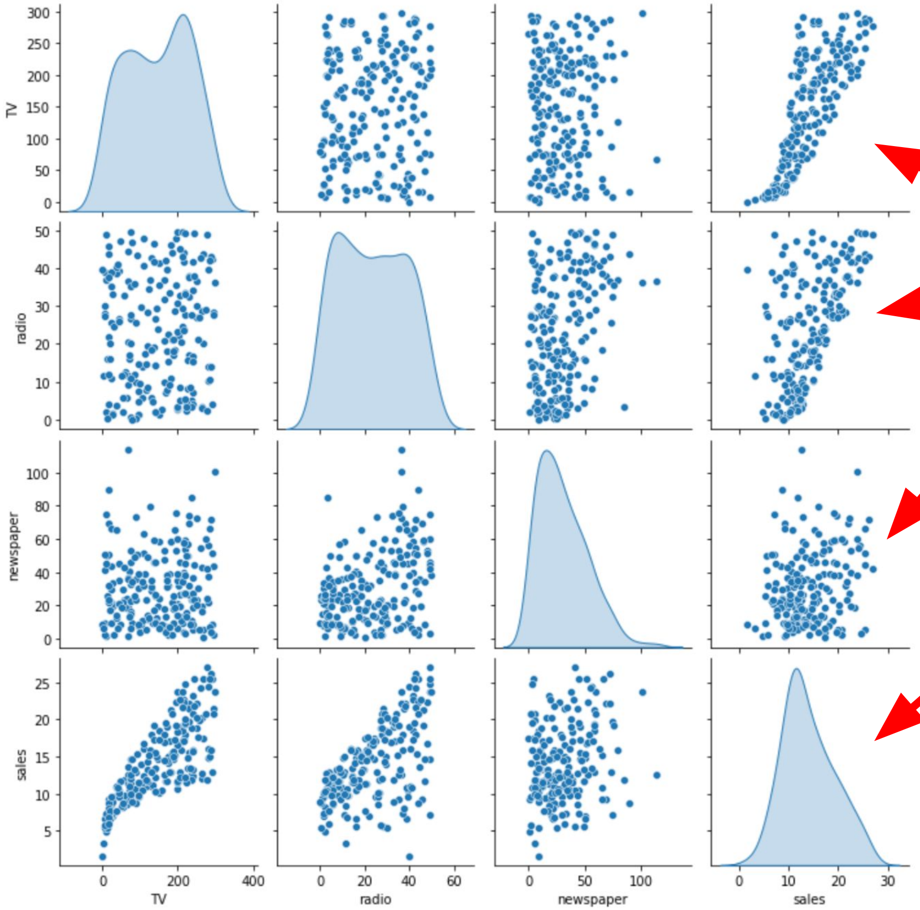
diag_kind parameter → This parameter refers to the diagonal plot in the grid (The correlation between the column to itself (TV expanse vs TV expanse).

We can choose from the following options:

- **hist (default)** → The diagonal plots are rendered as histogram.
- **kde** → The diagonal plots are rendered as kernel density estimate plots.
- **None** → No plot is rendered on the diagonal subplots.

Supervised Learning Framework - Python Example

```
In [192]: # Relationships between features
          sns.pairplot(df,diag_kind='kde')
Out[192]: <seaborn.axisgrid.PairGrid at 0x216014fb648>
```



The pairplot() show any combination between two columns in the dataframe provided

The diagonal plot generated as kde plot

ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

Supervised Learning Framework - Python Example

After we finished the exams our data, we now need to split the dataframe into **train set** and **test set.**

train_test_split() → The train_test_split() function from sklearn package allowing us to easily split our dataframe into 2 parts, <u>the train section</u> and <u>the test section</u>.

The function will return a tuple with the following datasets (X_train, X_test, y_train, y_test).

The function takes the following parameters:

- **X** → The features dataframe (containing only the features columns and their values)
- **y** → The label dataframe (containing only the label column and it's values)
- **test_size** → Decimal number between 0 - 1 representing the percentage of rows that should be taken to the test set (For example, 0.3 meaning 30% of the rows will be at the test section). train_test_split() <u>choose the rows randomly</u> in case the dataframe rows are ordered.
- **random_state** → Allowing us to determine the seed which will generate the random rows that will be selected to the test set (similar to random.seed() function).

Supervised Learning Framework - Python Example

Let's generate our X (features) and y (label) dataframes and execute the train_test_split():

```
In [6]: X = df.drop('sales', axis=1)
        y = df['sales']

        from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

If we will print the size of our original dataframe and the sizes of the X_train and X_test dataframes, we will see that the X_train size is 70% and our X_test size is 30% of our total dataframe

```
In [13]: print(len(df))
         print(len(X_train))
         print(len(X_test))

         200
         140
         60
```

Supervised Learning Framework - Python Example

In addition, The X_train rows index is matching to the y_train rows index so it will be easy for the model to match between the feature values and the corresponding label value.

```
In [15]: print(X_train.head(5))
         print()
         print(y_train.head(5))

            TV  radio  newspaper
169  284.3   10.6        6.4
97   184.9   21.0       22.0
31   112.9   17.4       38.6
12    23.8   35.1       65.9
35   290.7    4.1        8.5

169    15.0
97     15.5
31     11.9
12      9.2
35     12.8
Name: sales, dtype: float64
```

Now that we executed the train_test_split() we can start the model training and testing:

```
In [ ]: from sklearn.linear_model import LinearRegression
        model = LinearRegression()
        model.fit(X_train, y_train)
        model.predict(X_test)
```

We choose the LinearRegression model

We train our model on the train set data

We see the model results on the test set data

ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

Supervised Learning Framework - Python Example

When execute the predict() method on the X_test data set we will get an array with all the corresponding model predictions.

```
Out[16]: array([16.5653963 , 21.18822792, 21.55107058, 10.88923816, 22.20231988,
                13.35556872, 21.19692502,  7.35028523, 13.27547079, 15.12449511,
                 9.01443026,  6.52542825, 14.30205991,  8.97026042,  9.45679576,
                12.00454351,  8.91549403, 16.15619251, 10.29582883, 18.72473553,
                19.76821818, 13.77469028, 12.49638908, 21.53501762,  7.60860741,
                 5.6119801 , 20.91759483, 11.80627665,  9.08076637,  8.51412012,
                12.17604891,  9.9691939 , 21.73008956, 12.77770578, 18.1011362 ,
                20.07590796, 14.26202556, 20.93826535, 10.83938827,  4.38190607,
                 9.51332406, 12.40486324, 10.17045434,  8.09081363, 13.16388427,
                 5.2243552 ,  9.28893833, 14.09330719,  8.69024497, 11.66119763,
                15.71848432, 11.63156862, 13.35360735, 11.1531472 ,  6.33636845,
                 9.76157954,  9.4195714 , 24.25516546,  7.69519137, 12.15317572])
```

So we managed to train our model on a given data set, and see how it's perform on a test data set that we already know it's corresponding label value.

This allow us to easily test our model performance and decide if it's precise enough for our need or do we need to improve it.

ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

Class Exercise - Supervised Learning Framework

Instructions:

For this exercise use **'Store_Sales_Data.csv'** and implement the following instructions:

- Investigate the provided data set and find for each column it’s min, max and mean value.
- Explore the correlation between each column and the ‘stroe_sales’ label. Decide which feature column has the best linear correlation with the label column.
- Explore the correlation between all possible columns pair combinations and decide if there are columns that depend on each other.
- Execute train_test_split and divide your data set into train data set (85% of the columns) and test data set (15% of the columns). Use 101 as the random seed number.
- Train a linear model on your train data set and predict the results using your test data set.
  Print your model predictions result array.

Class Exercise Solution - Supervised Learning Framework