



Dr. Christian Tsoungui Obama

ML Cheat sheet #1

Your practical 7-step guide from business idea to a deployable regression model

A Foundational Guide for Aspiring Data Scientists

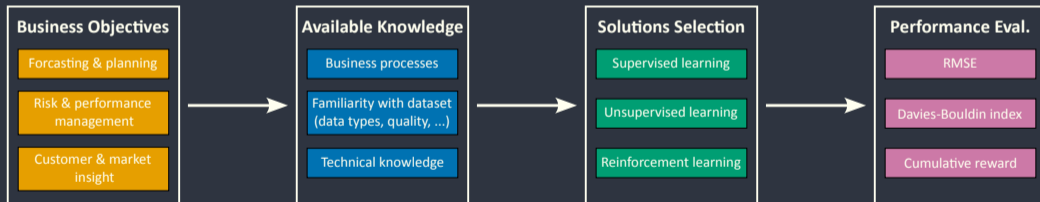
Based on
"Hands-On Machine Learning with Scikit-learn, Keras & TensorFlow"
by

Aurélien Geron

Step 1: Properly Frame the Problem

- Imagine you've been collecting weather data with a small sensor. You wonder: "If I know the humidity, can I predict the temperature?"
- Now think of a store tracking its advertising budget and sales. Someone asks: "If we change our ad budget, can we predict our sales?"

Building machine learning (ML) models for such questions requires proper problem framing.



Machine learning project roadmap: illustration of key points to consider when framing the problem

Practical steps to build regression models are described in what follows.

Step 2 : Acquire the data

Possible data sources:

- first-party data (e.g., company databases, operational data)
- online sources (e.g., web scraping, APIs)
- public, open data repository (e.g., Kaggle, UCI)

```
1 import kagglehub
2 from kagglehub import KaggleDatasetAdapter
3
4 # Loading a DataFrame from Kaggle
5
6 adsSales = kagglehub.dataset_load(
7     KaggleDatasetAdapter.PANDAS ,
8     "thorgodofthunder/tvradionewspaperadvertising/versions/1",
9     "Advertising.csv",
10 )
```

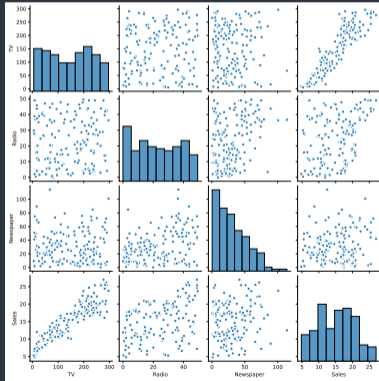
Example code to import "Advertising" dataset from Kaggle¹

¹www.kaggle.com/datasets/thorgodofthunder/tvradionewspaperadvertising

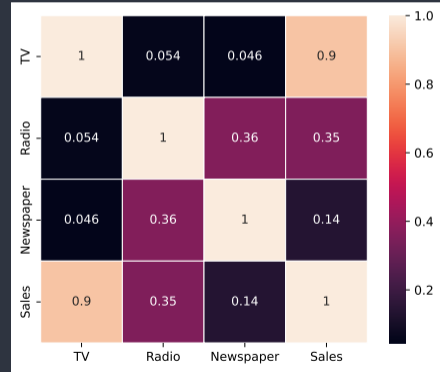
Step 3: Explore the data

Exploratory data analysis (EDA)

1. split data into train & test sets
2. investigate linear correlation (numeric attributes)
3. transform data using pipelines (e.g., imputation, one-hot-encoding)



Scatter plot matrix: point cloud resembles a straight line for linearly correlated attributes



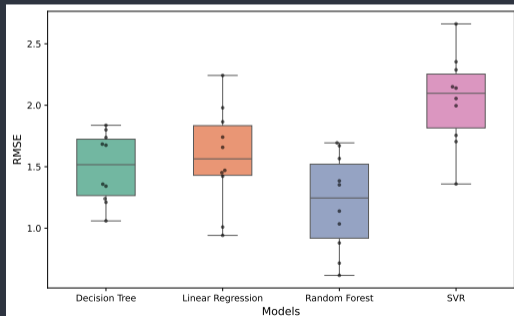
Correlation matrix: values quantify strength and direction of linear correlation

Step 4: Train & evaluate candidate models

Some candidate models for regression:

Models	Advantages	Disadvantages
Linear regression	simple, interpretable, fast to train	assumes linearity, sensitive to outliers & multicollinearity
Decision tree regressor	models non-linearity, easy to visualize & interpret	prone to overfitting, sensitive to data change & skewness
Support vector regression (SVR)	models non-linearity, effective for high-dimensional & medium-sized data	Computationally expensive for large datasets, difficult to interpret & sensitive to kernel choice

```
1 # Training a linear regression model
2
3 from sklearn.linear_model import LinearRegression
4 from sklearn.metrics import mean_squared_error
5 import numpy as np
6
7 lin_reg = LinearRegression()
8 lin_reg.fit(adsSales_prepared, adsSales_labels)
9
10 adsSales_predictions = lin_reg.predict(
11     adsSales_prepared)
12
13 lin_mse = mean_squared_error(adsSales_labels,
14     adsSales_predictions)
15
16 lin_rmse = np.sqrt(lin_mse)
```



Models performance: root mean squared scores (smaller is better)

Step 5 : Fine tune best candidate model (Random forest)

Model hyperparameters are fine-tuned to improve its performance:

1. explore large space with Random search
2. fine tune search with Grid search
3. use cross-validation in both cases

```
1 # Hyper-parameters tuning using grid search
2 from sklearn.model_selection import GridSearchCV
3
4 param_grid = [
5     {'n_estimators': [3, 10, 30, 50], 'max_features': [6, 8, 10]},
6     {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]},
7 ]
8
9 forest_reg = RandomForestRegressor()
10 grid_search = GridSearchCV(forest_reg, param_grid, cv=5,
11                             scoring='neg_mean_squared_error',
12                             return_train_score=True)
13
14 # Fit model for all combinations of parameters
15 grid_search.fit(adsSales_prepared, adsSales_labels)
16
17 # Picking best model from grid search
18 final_model = grid_search.best_estimator_
```

Step 6: Assess generalization of fine tuned model on test set

Prepare test set with same pipeline as the training set

```
1 # Defining test dataset
2 X_test = strat_test_set.drop("Sales", axis=1)
3 y_test = strat_test_set["Sales"].copy()
4
5 X_test_prepared = full_pipeline.transform(X_test)
```

Evaluate model performance


```
1 # Running predictions on test set
2 final_predictions = final_model.predict(X_test_prepared)
3
4 final_mse = mean_squared_error(y_test, final_predictions)
5 final_rmse = np.sqrt(final_mse)
```

Step 7: Save model for possible deployment

```
1 # Saving the model as joblib file
2
3 def save_model(model, name="model", model_path="./"):
4     """
5     Save the model with a given name to a specific location.
6
7     Parameters
8     -----
9     model :
10         The model trained.
11     name:
12         The file name with which to save the model
13     model_path:
14         The path in which the model is saved
15         Example:
16             "./models/"
17
18     """
19     os.makedirs(model_path, exist_ok=True)
20     model_name = name + ".pkl"
21     pkl_path = os.path.join(model_path, model_name)
22     joblib.dump(model, pkl_path)
23
24 MODEL_PATH = os.path.join("models", "adsSales")
25 save_model(final_model, name="Random_forest_V1.1", model_path=MODEL_PATH)
```




Did you find this useful?

 **Get full notebook:** » [GitHub repository](#) «

 **Comment below:** How do you choose your candidate models for regression?

 **Follow me:** for the next cheat sheet in this series.

 **Like this post:** to help it reach more people.

Let's Connect!



/in/christiantsoungui



@chrisTs