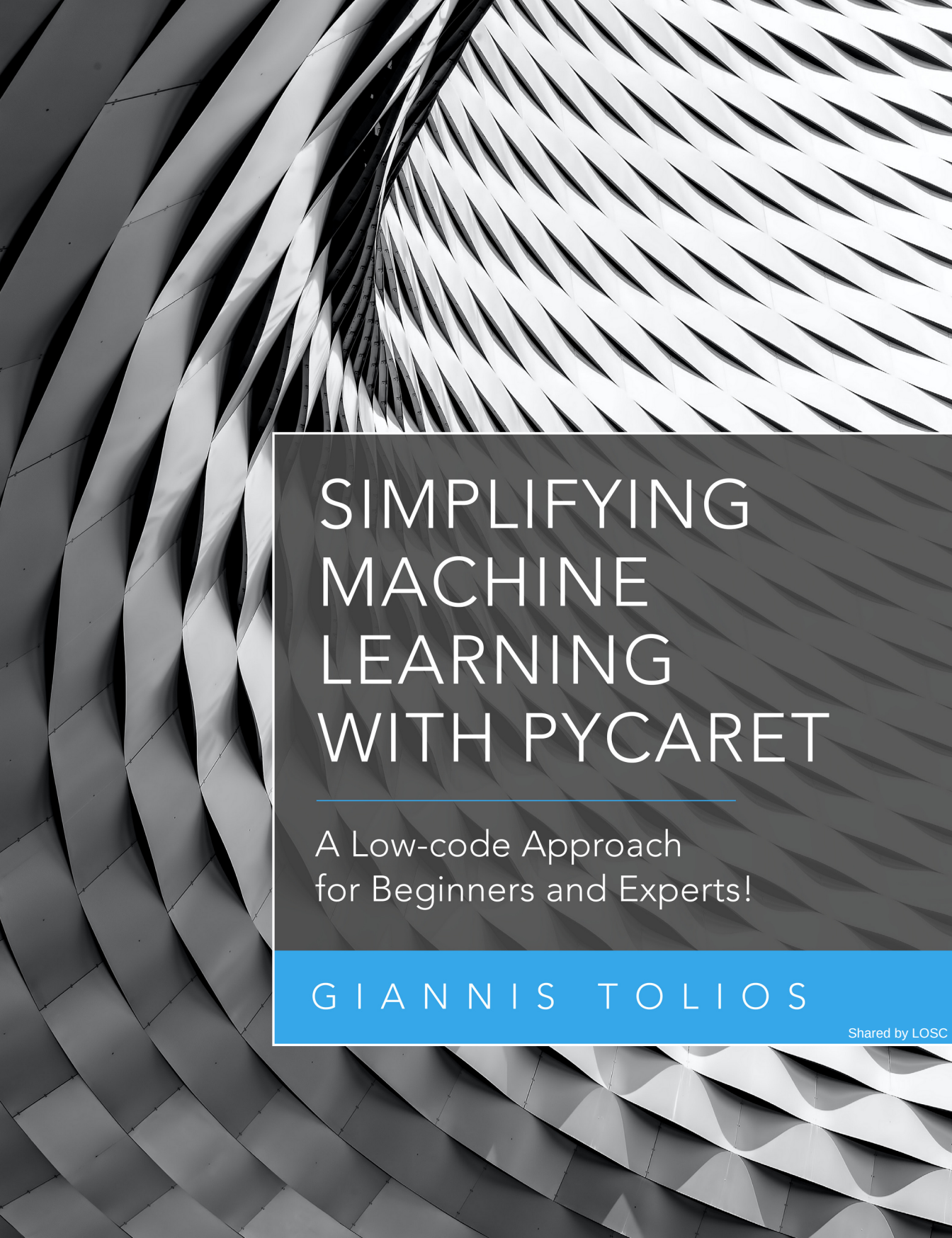


SIMPLIFYING MACHINE LEARNING WITH PYCARET

A Low-code Approach for Beginners and Experts!

GIANNIS TOLIOS

Simplifying Machine Learning with PyCaret

A Low-code Approach for Beginners and Experts!

Giannis Tolios

This book is for sale at http://leanpub.com/pycaretbook

This version was published on 2021-10-20

Contents

1. Preface

I decided to write this book for a couple of reasons. After publishing articles on websites like Towards Data Science, I was interested in taking a step forward as a writer. In the beginning of 2021, a major international publisher proposed me to collaborate on a PyCaret book, as I had already written an article about that library. At first I was interested, but eventually decided to publish my book on Leanpub, as it made more sense from a financial perspective. Furthermore, at the time of writing there is no other PyCaret book available, while there seems to be significant demand for it.

In this book, I will present an overview of all the main features of PyCaret, by focusing on practical case studies that are easy to follow. After reading it, you'll be able to create and deploy machine learning models, thus taking advantage of this powerful technology. Machine learning is a field that has grown substantially in the past years, due to technological and scientific advancements. Data scientists and machine learning engineers are among the best paid professionals in the modern job market, so learning those skills is beneficial for every developer. The low-code approach of PyCaret makes it suitable for beginners, as well as seasoned developers that want to get results quickly, thus increasing their productivity.

The following chapters contain hands-on tutorials for the main machine learning modules that are included with PyCaret, such as regression, classification, anomaly detection and clustering. Furthermore, I will also explain how to deploy ML models as web applications, based on the Streamlit library and the associated Streamlit Cloud service. This will let you share your work with others, thus showcasing your skill and expertise. I hope you enjoy reading this book, and I encourage you to share your thoughts and feedback[1] with me!

[1] https://forms.gle/1hsbBtG1ZSEcyrH27

2. About the Author

Giannis Tolios is a data scientist who is passionate about expanding his knowledge and evolving as a professional. He has collaborated with numerous companies worldwide as a freelancer, and completed projects related to machine learning, NLP, time series forecasting, scientific data visualization and others. Giannis also enjoys writing about data science[1] at established websites such as Towards Data Science and Analytics Vidhya. Giannis strongly believes that technology should be used for good, and is constantly looking for new ways to help mitigate challenges like climate change and economic inequality, by using data science. If you want to learn more about Giannis, you can visit his personal website[2], or follow him on LinkedIn[3], where he's regularly posting content about data science and other topics.

[1] https://giannistolios.medium.com/
[2] https://giannis.io
[3] https://www.linkedin.com/in/giannis-tolios

3. About the Book

Who this Book is for

This book is targeted towards Python developers that want to familiarize themselves with the PyCaret library, as well as machine learning in general. The content is beginner-friendly, so I assume no prior knowledge of machine learning techniques and methods. Experienced data scientists and machine learning engineers can also benefit from reading this book, as it will help them acquaint themselves with the low-code approach of PyCaret, and improve their workflow. This book will focus on practical coding examples, so I will only provide basic explanations of theoretical concepts. In case you want to dig deeper into the theory, I suggest that you start with An Introduction to Statistical Learning[1], an excellent book that is provided as a free PDF download by the authors.

Prerequisites

You should have some basic knowledge of Python 3, linear algebra, statistics and probability theory. The math included in this book will be minimal, as it isn't supposed to be a college textbook, but rather a practical guide for developers. Regardless, having a solid understanding of some fundamental mathematical concepts is important. The following resources should be helpful to beginners and people who want to freshen up their math skills.

- The Official Python Tutorial[2]
- Free Linear Algebra Course on Khan Academy[3]
- Free Statistics and Probability Course on Khan Academy[4]

[1] https://www.statlearning.com/
[2] https://docs.python.org/3/tutorial/index.html
[3] https://www.khanacademy.org/math/linear-algebra
[4] https://www.khanacademy.org/math/statistics-probability/

Software Requirements

The code in this book should work on all major operating systems, i.e. Microsoft Windows, Linux and Apple macOS. You will need to have Python 3 installed on your computer, as well as JupyterLab. I suggest that you use Anaconda[5], a machine learning and data science toolkit that includes numerous helpful libraries and software packages. Anaconda can be freely downloaded at this link[6]. Alternatively, you can use a cloud service like Google Colab[7] to run Python code without worrying about installing anything on your machine.

Installing PyCaret

The PyCaret library can be installed locally by executing the following command on your Anaconda terminal. You can also execute the same command on Google Colab or a similar service, to install the library on a remote server.

```
pip install pycaret[full]==2.3.4
```

After executing this command, PyCaret will be installed and you'll be able to run all code examples of the book. It is recommended to install the optional dependencies as well, by including the `[full]` specifier. Furthermore, installing the correct package version ensures maximum compatibility, as I used PyCaret ver. 2.3.4 while working on this book. Finally, creating a conda environment[8] for PyCaret is considered to be best practice, as it will help you avoid conflicts with other packages and make sure you always have the correct dependencies installed.

[5] https://www.anaconda.com/
[6] https://www.anaconda.com/products/individual
[7] https://colab.research.google.com/
[8] https://docs.conda.io/projects/conda/en/latest/user-guide/getting-started.html#managing-envs

Using JupyterLab

JupyterLab is a powerful web-based user interface that lets you create and execute Jupyter notebooks containing Python code. You can run JupyterLab by executing the following command on the Anaconda terminal. Substituting `lab` with `notebook` will run the older Jupyter Notebook web interface, but I only recommend it if you have problems or compatibility issues with JupyterLab.

```
jupyter lab
```

When JupyterLab starts, you can create a new Jupyter notebook and run the code that is provided in each chapter of the book. After executing a Jupyter notebook, both the code and its output are displayed to the user. This file format can also contain text, figures and visualizations. Jupyter notebook is a powerful format that has become the standard in data science and machine learning. You can refer to the official JupyterLab Documentation[9] for more information and help about using JupyterLab. In the following chapters, each code snippet represents a single Jupyter notebook cell that can be executed by pressing the Shift + Enter keyboard shortcut.

Github Repository

You can access all code examples included in this book at the official Github repository[10], with each folder containing the Python code of the same-titled book chapter. You can also clone the repository to your local machine if you wish. Git is the de facto standard for version control, so I assume that most developers will be familiar with it. In case you haven't used it, please refer to the official quick reference guide[11].

[9] https://jupyterlab.readthedocs.io/en/stable/index.html
[10] https://github.com/derevirn/pycaret-book
[11] https://training.github.com/downloads/github-git-cheat-sheet/

4. A Brief Introduction to Machine Learning

Before starting a machine learning project, you need to have a solid grasp of the theory behind that technology. This book is targeted mostly to beginners and Python developers that want to improve their skill set and familiarize themselves with PyCaret, so I assume no prior knowledge of machine learning. Therefore, I will provide a brief introduction to the basic theoretical concepts that are necessary to understand the rest of the book. In case you are familiar with them, feel free to skip this chapter.

What is Machine Learning?

The term machine learning was popularized by Arthur Samuel, a prominent figure in the field of artificial intelligence[1]. Samuel defined machine learning as the field of study that gives computers the ability to learn without being explicitly programmed. This was a revolutionary approach, as traditional computer algorithms include all the necessary steps that have to be executed, explicitly defined by a human programmer. Unfortunately though, some problems are simply too difficult and complex, making it almost impossible to define an algorithm for them. Machine learning can be used to circumvent that obstacle by training the computer to find the solution, rather than giving it a set of explicit instructions. You may see machine learning being used interchangeably with the terms artificial intelligence and deep learning. Although associated, those terms aren't identical. It is generally accepted that machine learning is a subfield of artificial intelligence, while deep learning is a subfield of machine learning.

Machine Learning Categories

Machine learning is typically divided in three main categories, i.e. supervised learning, unsupervised learning and reinforcement learning[2]. I will give a brief description for each one of those categories, as well as some real-life examples so you can easily get a grasp of them.

Supervised Learning

The main goal in supervised learning is to create a predictive model based on a set of labeled instances, known as the training dataset[3]. This dataset is a matrix where rows represent the instances, while columns represent the features and label. After the model is successfully trained, it can be used to make predictions on unlabeled data. Supervised learning is the most common type of machine learning, including tasks such as regression and classification. It can be applied to various kinds of data, like structured/tabular, text, image and video.

To help you understand supervised learning better, I am going to describe a simple regression example, where the goal is to predict a continuous value. In this case, the training dataset includes cars with attributes like dimensions, horsepower, engine size, number of cylinders and gas mileage. Those attributes are going to be used as features, while price is considered to be the label. After training a regression model on that dataset, it will be able to estimate the price of a car based on the rest of its attributes. We are going to examine a complete case study of regression in the next chapter.

Let's also consider a classification example, where the goal is to predict a categorical class label. The dataset is comprised of patient records, including attributes such as age, sex, resting blood pressure, chest pain type and maximum heart rate. Those attributes are the features, while the presence or absence of heart disease in each patient is the label. This is known as a binary classification task, because there are only two classes. The trained model, known as a classifier in this case, will be able to evaluate whether a patient is at risk of having heart disease. Classification will be covered in chapter 3 of this book.

Unsupervised Learning

In case we can't get a dataset with labeled instances, unsupervised learning algorithms can help us discover hidden structures on our data. Unsupervised learning includes tasks such as clustering and anomaly detection. Those algorithms can be applied to structured data, text, image and other types of data, similarly to supervised learning.

Let's consider the following clustering example. Suppose that a company has a dataset with information about its customers, including various attributes, such as age, income and spending behavior. Groups of customers with similar characteristics can be created with the help of a clustering algorithm, in a process known as market segmentation. That information can be used to create effective marketing campaigns for each group, thus improving company sales and increasing revenue. We are going to examine a clustering case study in chapter 4.

Anomaly detection can be used in numerous different cases, such as fraud in financial transactions. In this case, every transaction is analyzed by an anomaly detection algorithm to detect potential fraud cases. Fraudulent transactions may include various possible events, like stolen credit cards being used for purchases, or hackers attempting to transfer funds illegally. Every suspicious transaction will be flagged as such by the algorithm. Anomaly detection will be studied in chapter 5.

Reinforcement Learning

The goal of reinforcement learning is to create an intelligent agent that interacts with its environment. That agent gets rewarded for every correct decision it makes, thus learning to perform various actions via trial-and-error[4]. Reinforcement learning can be used to create autonomous vehicles such as self-driving cars[5]. It can also be used to improve the AI of video games, thus resulting in more immersive and realistic experiences for gamers. We are not going to examine reinforcement learning in detail, as PyCaret doesn't support it currently.

5. Regression

The Linear Regression Model

A fundamental task in supervised machine learning is regression, where the goal is to predict a continuous value. This is achieved by understanding the relationship between the target variable y and the feature variables x on a given dataset. One of the most basic regression models is Linear Regression[6], which is defined in the following equation. The equivalent vectorized form of the equation is also provided, where the inner product of the transposed vector $\boldsymbol{\beta}^\top$ and $\boldsymbol{X_n}$ is calculated.

$$y_n = \beta_0 + \beta_1 x_{n1} + \cdots + \beta_p x_{np} + \epsilon_n = \boldsymbol{\beta}^\top \boldsymbol{X_n} + \epsilon_n$$

- y_n is the target variable for the nth instance of the given dataset.
- x_1 to x_p are the feature variables.
- β_0 is the intercept term.
- β_1 to β_p are the coefficients of the feature variables.
- ϵ is the error variable.

Regression with PyCaret

Besides Linear Regression, there are numerous other models available, such as Lasso[7], Random Forest[8], Support Vector Machines[9] and Gradient Boosting[10]. In the rest of this chapter, we are going to see how PyCaret can help us choose and train the optimal regression model for a specific dataset. We are also going to learn about Exploratory Data Analysis (EDA), a method that lets you examine and understand the basic statistical properties of a dataset.

Importing the Necessary Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
from pycaret.datasets import get_data
from pycaret.regression import *
mpl.rcParams['figure.dpi'] = 300
```

First of all, we import the Python libraries that are necessary for our project. Some standard machine learning libraries are included, such as pandas, Matplotlib and Seaborn. Furthermore, we import all PyCaret functions that are related to regression. The last line specifies that Matplotlib figures will have a 300 DPI resolution, but you can omit that if you wish.

Loading the Dataset

Machine learning projects can only succeed if the proper data are available, so PyCaret includes a variety of datasets that can be used to test its features. In this chapter, we are going to use `insurance.csv`, a dataset that originates from the book Machine Learning with R, by Brett Lantz[11]. This is a health insurance dataset, where the features are various attributes including age, sex, Body Mass Index (BMI)[1], whether the person is a smoker or not, number of children and US region. Furthermore, the dataset target variable is the billed charges for every individual. Real-world data are usually more complex, but working with so-called toy datasets will help you grasp the concepts and techniques before dealing with more difficult cases.

[1] https://en.wikipedia.org/wiki/Body_mass_index

```
data = get_data('insurance')
```

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

We use the `get_data()` PyCaret function to load the dataset to a pandas dataframe. The output is equivalent to the `head()` pandas function that prints the first 5 dataset rows. This lets us get a first glimpse of the data we are working with.

```
data.info()
```

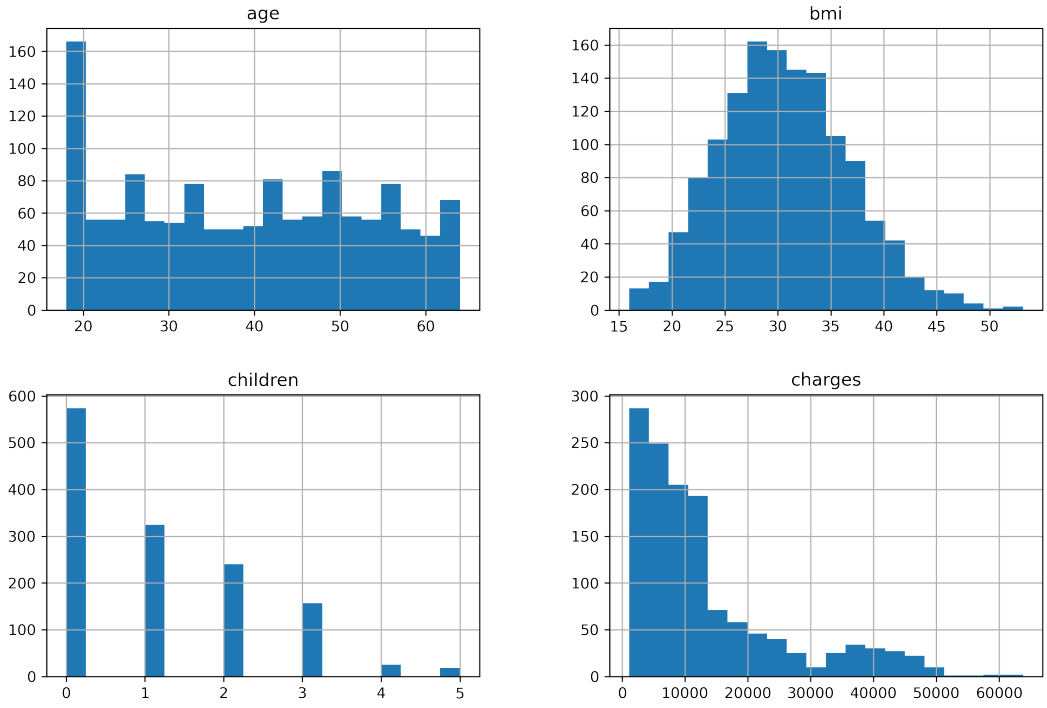
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1338 non-null   int64
 1   sex       1338 non-null   object
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
 4   smoker    1338 non-null   object
 5   region    1338 non-null   object
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

We use the pandas `info()` function to examine some basic information about the dataset. As we can see, there are 1338 rows and none of the columns have null values. Furthermore, the data type of each column has been automatically inferred by the pandas library.

Exploratory Data Analysis

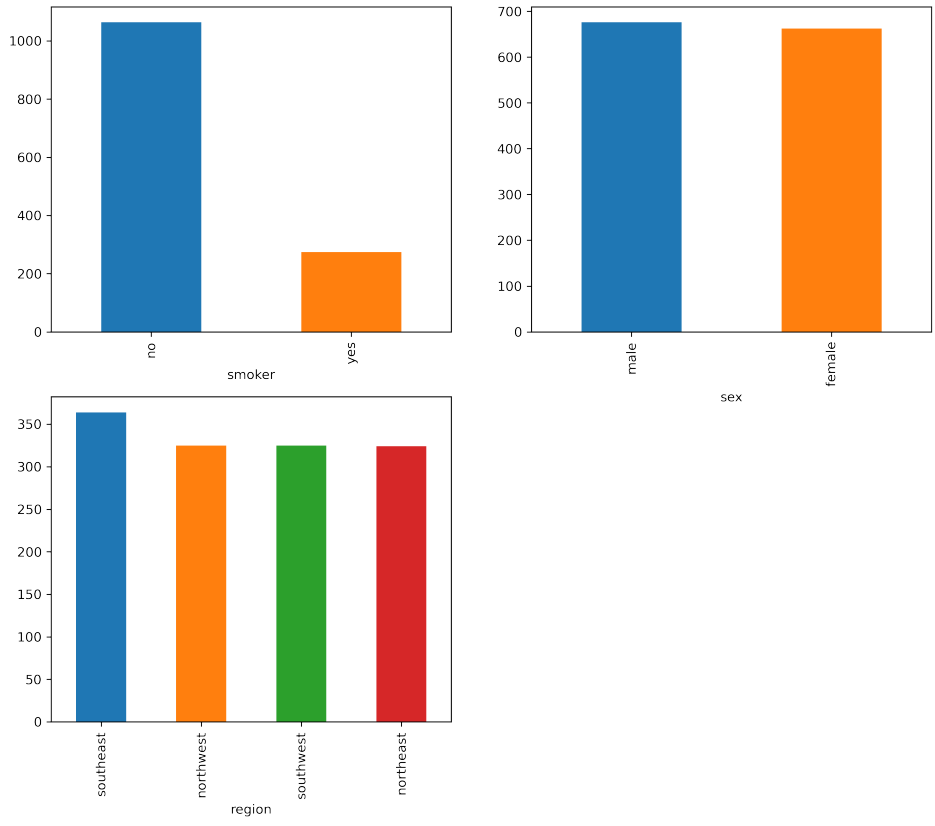
We are now going to perform Exploratory Data Analysis (EDA) on our data. As mentioned earlier, EDA is a method that helps us understand the dataset properties by using descriptive statistics and visualization. It is an important part of every machine learning or data science project, as you have to understand the dataset before being able to utilize it.

```
numeric = ['age', 'bmi', 'children', 'charges']

data[numeric].hist(bins=20, figsize = (12,8))
plt.show()
```



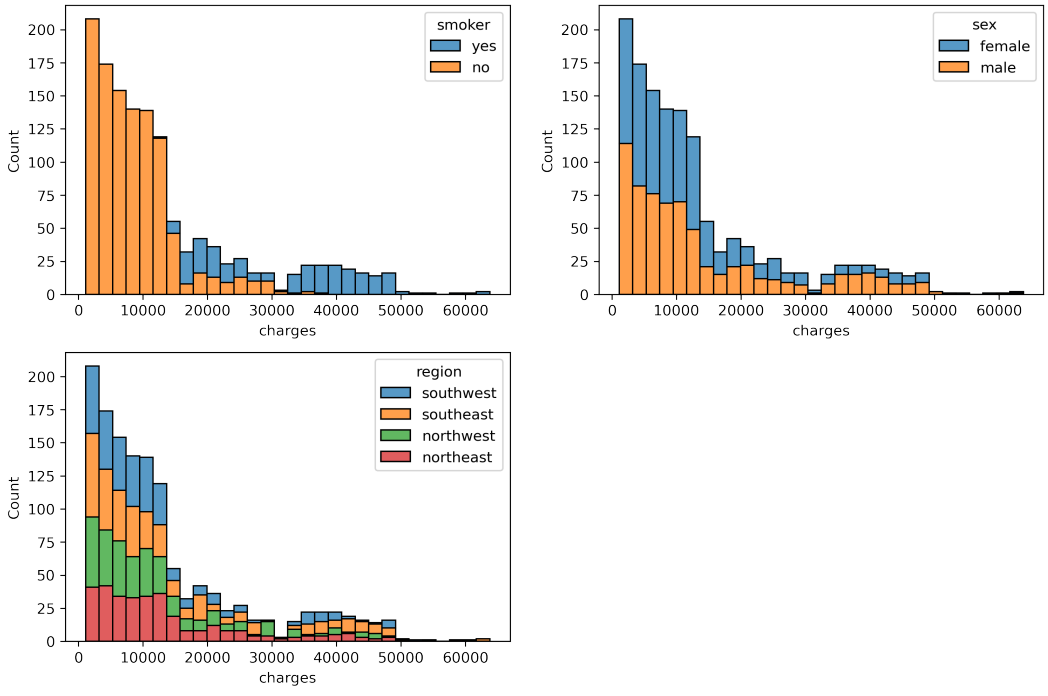
The distribution of numeric variables can be visualized with a histogram that can be easily created with the `hist()` pandas function. It is obvious that some of the variables have right-skewed distributions that may cause problems with regression models, so we'll have to deal with that later.

```
categorical = ['smoker', 'sex', 'region']
color = ['C0', 'C1', 'C2', 'C3']
fig, axes = plt.subplots(2, 2, figsize = (12,10))
axes[1,1].set_axis_off()

for ax, col in zip(axes.flatten(), categorical) :
    data[col].value_counts().plot(kind = 'bar', ax = ax, color = color)
    ax.set_xlabel(col)
```



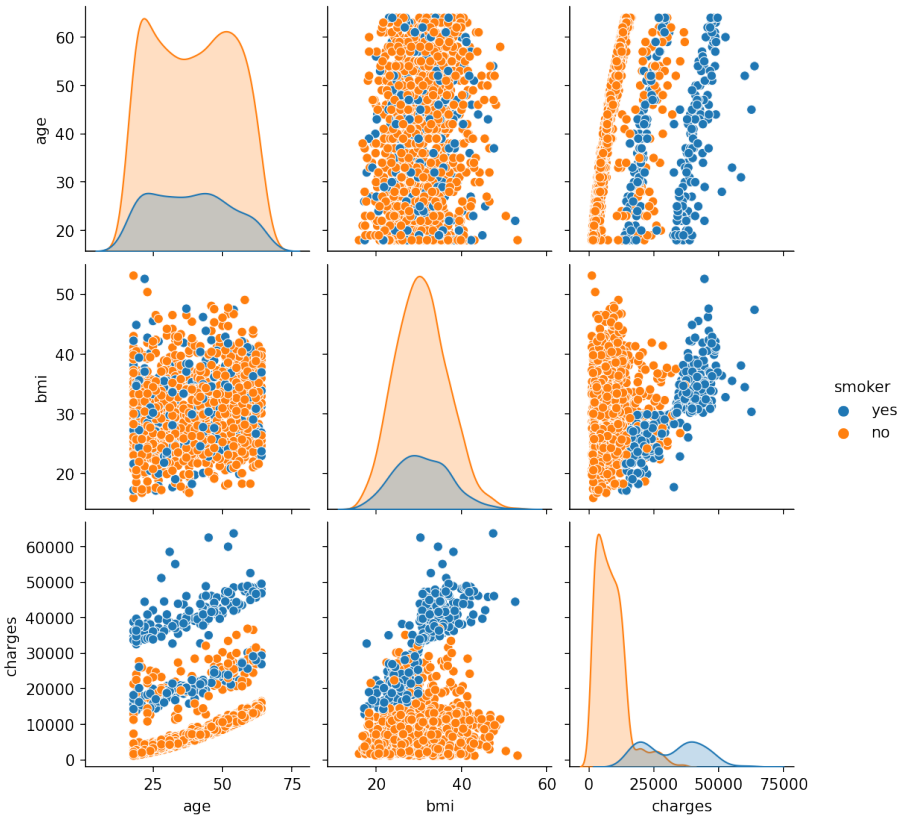
Using bar charts is the standard way of plotting categorical variables. We can accomplish that easily, by using the `value_counts()` and `plot()` pandas functions. As we can see, the smoker variable has uneven distribution, with only 20% of people being smokers. On the other hand, the sex and region variables are equally distributed.

```
fig, axes = plt.subplots(2, 2, figsize=(12,8))
axes[1,1].set_axis_off()

for ax, col in zip(axes.flatten(), categorical):
    sns.histplot(data, x='charges', hue=col, multiple='stack', ax=ax)
```



The `histplot()` Seaborn function lets us visualize the relationship between numeric and categorical variables by using hue mapping. In this case, we plot the target variable histogram, colored differently for every category of the smoker, sex and region variables. Evidently, smokers get significantly higher charges compared to non-smokers. This is expected, as the health risks associated with smoking are numerous and well-documented.

```
cols = ['age', 'bmi', 'charges', 'smoker']
sns.pairplot(data[cols], hue='smoker')
plt.show()
```



Scatter plots are a type of visualization that helps us understand the relationship between numeric variables. The `pairplot()` Seaborn function creates a grid of scatter plots for all pairs of numeric variables in a given dataset. The diagonal contains distribution plots of the variables, such as histograms or KDE plots in this case. Once again, we use hue mapping to highlight the differences between smokers and non-smokers. As we can see, age is correlated with charges, i.e. people get higher charges as they grow older. In spite of that, being a non-smoker keeps the cost lower for most people, regardless of their age. Furthermore, overweight and obese people don't seem to get significantly higher charges, unless they are also smokers.

Initializing the PyCaret Environment

```
reg = setup(data=data, target='charges', train_size = 0.8, session_id = 7402,
            numeric_features = numeric[:-1], categorical_features = categorical,
            transformation = True, normalize = True, transform_target = True)
```

	Description	Value
0	session_id	7402
1	Target	charges
2	Original Data	(1338, 7)
3	Missing Values	False
4	Numeric Features	3
5	Categorical Features	3
6	Ordinal Features	False
7	High Cardinality Features	False
8	High Cardinality Method	None
9	Transformed Train Set	(1070, 9)
10	Transformed Test Set	(268, 9)
11	Shuffle Train-Test	True
12	Stratify Train-Test	False
13	Fold Generator	KFold

After the EDA part of the project is complete, the next step is to initialize the PyCaret environment. We can accomplish that by using the `setup()` function, which prepares the data for model training and deployment. This function has numerous parameters, but we are only going to focus on the most important. In case you want to delve deeper, you can refer to the documentation page of the PyCaret Regression module[2].

[2] https://pycaret.readthedocs.io/en/latest/api/regression.html

After running the `setup()` function, a table with its parameters and settings is printed, as seen in the image. We are now going to examine the preprocessing pipeline that has been applied to the dataset.

Identifying Numeric and Categorical Features

PyCaret can automatically infer whether a feature is numeric or categorical. When you execute the `setup()` function, you will be prompted to verify that the features have been identified correctly. Alternatively, you can specify which features are categorical or numeric using the `categorical_features` and `numeric_features` parameters, as we did in this case.

Train/Test Split

Splitting a dataset to a train and test subset is standard practice in machine learning, because it is important to evaluate model performance on data that haven't been used to train it. In this case, we have set the `train_size` parameter to 0.8, meaning that the machine learning model will be trained on 80% of the original data, while the 20% will be used for testing purposes.

Normalization of Numeric Features

Some regression models require numeric features to be normalized by having $\mu = 0$ and $\sigma = 1$. We enabled normalization by setting the `normalize` parameter to `True`, thus applying the standard scaler. This method replaces each value of the feature with the z-score, which is defined as $z = \frac{x-\mu}{\sigma}$. Changing the `normalize_method` parameter lets us choose a different normalization method, with other options including the Min Max scaler and the Robust scaler.

Transformation of Numeric Features and Target

In the EDA part of the project, we realized that some of the features and the target variable are skewed. This can degrade the performance of some regression models, as they are designed to work with variables that have a normal distribution. We can deal with that problem by transforming the variables so their distribution is closer to normal. Transformation was enabled by setting the `transformation` and `transform_target` parameters to `True`.

One-Hot Encoding Categorical Features

Categorical features are supported by some regression models, but they can cause problems with others. It is therefore advised to apply a categorical encoding method, thus converting them to numeric variables. PyCaret applies one-hot encoding by default, where new binary features are created for each category of the categorical features.

Printing the Preprocessed Features

```
get_config('X').head()
```

	age	bmi	children	sex_female	smoker_no	region_northeast	region_northwest	region_southeast	region_southwest
0	-1.462763	-0.408971	-1.053884	1.0	0.0	0.0	0.0	0.0	1.0
1	-1.535749	0.543953	0.203960	0.0	1.0	0.0	0.0	1.0	0.0
2	-0.807782	0.426243	1.414612	0.0	1.0	0.0	0.0	1.0	0.0
3	-0.445654	-1.350171	-1.053884	0.0	1.0	0.0	1.0	0.0	0.0
4	-0.517962	-0.240519	-1.053884	0.0	1.0	0.0	1.0	0.0	0.0

This is what the feature variables look after the preprocessing pipeline has been applied to them. We can see that the numeric features have different values due to the normalization/transformation, and the categorical features have been converted to multiple binary variables due to the one-hot encoding.

Comparing Regression Models

```
best = compare_models(sort='RMSE')
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
gbr	Gradient Boosting Regressor	2275.4641	22815428.3119	4750.1089	0.8350	0.3858	0.1874	0.0190
rf	Random Forest Regressor	2342.1429	22959433.7357	4762.0337	0.8351	0.4097	0.2091	0.0760
ada	AdaBoost Regressor	3257.2171	23279230.3521	4807.0257	0.8339	0.4770	0.4264	0.0100
lightgbm	Light Gradient Boosting Machine	2491.6919	24030584.9610	4865.2118	0.8272	0.4153	0.2118	0.0590
catboost	CatBoost Regressor	2485.6847	25075332.3085	4978.9614	0.8189	0.4071	0.1994	0.5830
et	Extra Trees Regressor	2364.4206	25237906.1980	4999.4284	0.8167	0.4283	0.2116	0.0680
xgboost	Extreme Gradient Boosting	2931.6919	31946244.2000	5615.7612	0.7678	0.4551	0.2602	0.0920
dt	Decision Tree Regressor	3031.4152	42283353.7664	6468.0098	0.6936	0.5132	0.3181	0.0090
omp	Orthogonal Matching Pursuit	5645.3006	59119658.9118	7679.3608	0.5758	0.6831	0.6880	0.0080
ridge	Ridge Regression	4066.3602	61583198.0000	7714.4266	0.5583	0.4400	0.2707	0.0080
br	Bayesian Ridge	4072.9367	61948316.9816	7735.3075	0.5556	0.4399	0.2705	0.0080
lar	Least Angle Regression	4081.2537	62419165.3459	7762.5119	0.5521	0.4399	0.2702	0.0080
lr	Linear Regression	4081.2544	62419200.4000	7762.5147	0.5521	0.4399	0.2702	0.0080
knn	K Neighbors Regressor	4590.2544	70154126.3724	8271.6145	0.5162	0.5252	0.3131	0.0200
huber	Huber Regressor	4211.3096	80449305.4129	8799.2293	0.4214	0.4535	0.2076	0.0090
par	Passive Aggressive Regressor	5841.9890	94762106.9683	9607.9039	0.2863	0.6406	0.4609	0.0090
en	Elastic Net	8222.6688	160918303.2000	12608.0390	-0.1206	0.9079	0.9707	0.0080
llar	Lasso Least Angle Regression	8249.2145	161224210.7582	12619.7361	-0.1227	0.9107	0.9777	0.0080
lasso	Lasso Regression	8249.2145	161224220.8000	12619.7366	-0.1227	0.9107	0.9777	0.0080

As we mentioned earlier, there are numerous regression models available, and choosing the best one for our data can be challenging. The `compare_models()` function simplifies this process, by training all the available models and displaying a table with the results. The first column contains the regression model name, while the rest are various metrics. This table may seem intimidating if you are a beginner, but it actually isn't that complicated. As we can see, there are various metrics that can be used to evaluate the performance of a regression model, but we are going to focus on Root Mean Squared Error (RMSE), so we sorted the results based on it.

RMSE is one of the most widely used metrics for regression, and it is defined as the square root of the average of squared errors, between actual and predicted values. Lower RMSE values indicate a more accurate regression model, so in this case, the most accurate model is Gradient Boosting Regressor with an RMSE value of 4750.1089.

$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x_i})^2}$

Creating the Model

```
model = create_model('gbr', fold = 10)
```

	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	2324.8131	24607040.5795	4960.5484	0.8393	0.3810	0.1751
1	2571.2202	29114809.8137	5395.8141	0.7413	0.4376	0.2551
2	2483.0865	21498482.3051	4636.6456	0.8585	0.4261	0.2000
3	2545.2911	25489822.2086	5048.7446	0.8158	0.3937	0.1765
4	2218.7259	19926308.9440	4463.8894	0.8240	0.4237	0.2161
5	1991.9530	17546103.1066	4188.8069	0.9050	0.2883	0.1446
6	2557.6356	30138770.5115	5489.8789	0.8316	0.4120	0.1911
7	1904.3245	16102540.3261	4012.7971	0.8950	0.3101	0.1860
8	1789.5416	17459155.1593	4178.4154	0.8269	0.3299	0.1574
9	2368.0500	26271250.1647	5125.5488	0.8125	0.4560	0.1721
Mean	2275.4641	22815428.3119	4750.1089	0.8350	0.3858	0.1874
SD	274.1307	4766740.8468	501.8900	0.0435	0.0546	0.0297

We can now train the Gradient Boosting Regressor by using the `create_model()` function, which uses k-fold cross validation to evaluate model accuracy. The dataset is partitioned into k subsamples, with one of them being retained for validation, while the rest are used to train the model. This process is repeated k times, with the resulting metrics for each subsample being displayed when it's completed.

Tuning the Model

```
params = {'learning_rate': [0.01, 0.02, 0.05],
        'max_depth': [1,2, 3, 4, 5, 6, 7, 8],
        'subsample': [0.4, 0.5, 0.6, 0.7, 0.8],
        'n_estimators' : [100, 200, 300, 400, 500, 600]}

tuned_model = tune_model(model, optimize = 'RMSE', fold = 10,
                        custom_grid = params, n_iter = 100)
```

	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	2221.3251	23485165.0834	4846.1495	0.8466	0.3680	0.1698
1	2413.4985	28417256.3250	5330.7838	0.7475	0.4269	0.2226
2	2358.1966	20434514.3855	4520.4551	0.8655	0.4078	0.1862
3	2224.8093	22543132.0952	4747.9608	0.8371	0.3895	0.1590
4	1954.8899	17295001.5227	4158.7259	0.8472	0.4099	0.1875
5	1866.4875	16659066.2449	4081.5519	0.9098	0.2809	0.1412
6	2486.6686	29110527.1669	5395.4172	0.8374	0.4152	0.1857
7	1862.5828	15867899.1150	3983.4532	0.8966	0.2974	0.1748
8	1746.2978	16452845.2892	4056.2107	0.8368	0.3261	0.1598
9	2372.9397	26292453.2888	5127.6167	0.8124	0.4505	0.1706
Mean	2150.7696	21655786.0517	4624.8325	0.8437	0.3772	0.1757
SD	255.0557	4826683.3525	516.4401	0.0425	0.0546	0.0208

Hyperparameter tuning is a technique where the various parameters of a machine learning model are tweaked to optimize its performance. The `tune_model()` function can be used to apply hyperparameter tuning on a trained model by using the randomized search method, where a random sample of possible combinations is tested.

A dictionary with the hyperparameters of our preference was passed to the `custom_grid` parameter of the function. We can see that after tuning the model, the RMSE value decreased to 4624.8325. You should keep in mind that hyperparameter tuning can be a computationally intensive and time-consuming process. In case it takes too long to complete on your computer, you can try decreasing the number of iterations by setting a lower value on the `n_iter` parameter. Each regression model has a different set of hyperparameters that can be tuned, so you'll have to consult its documentation to select them. In this case, you can refer to the Gradient Boosting Regression[3] documentation page of scikit-learn[4].

Making Predictions

```
cols = ['age', 'bmi', 'children', 'sex_female', 'smoker_no','charges','Label']
predictions = predict_model(tuned_model)
predictions[cols].head()
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
0	Gradient Boosting Regressor	1937.1153	16816546.1060	4100.7982	0.8921	0.3297	0.1730

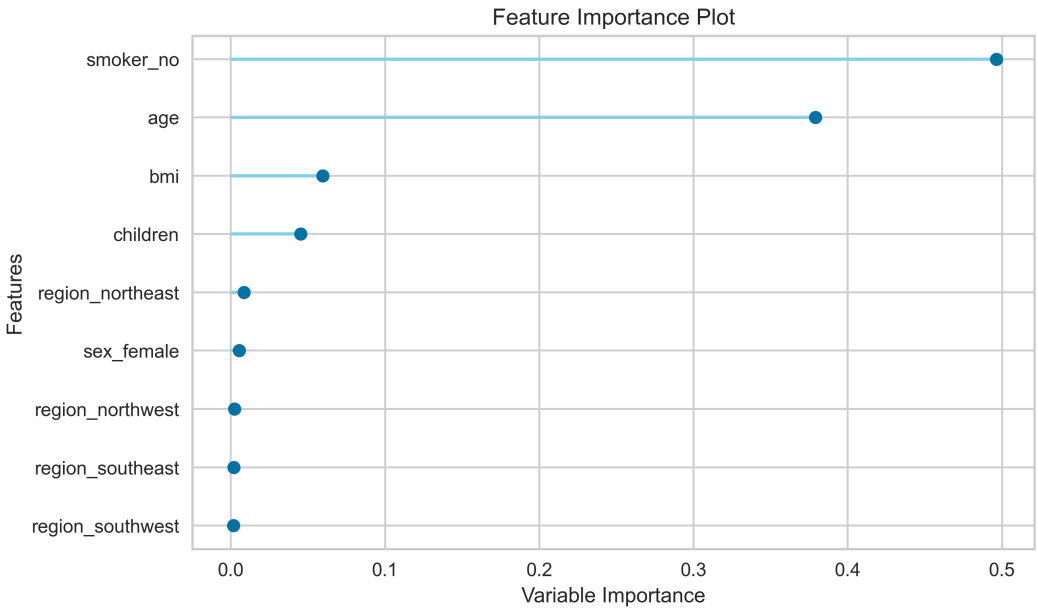
	age	bmi	children	sex_female	smoker_no	charges	Label
0	0.558673	-2.023842	0.203960	0.0	1.0	8627.541016	9174.750784
1	-0.880372	-0.739121	-1.053884	0.0	1.0	3070.808594	3816.227189
2	1.055905	-0.017651	-1.053884	0.0	1.0	10231.500000	12099.833376
3	1.197541	0.522682	-1.053884	0.0	0.0	43921.183594	43471.562081
4	0.273279	-0.619599	0.934173	1.0	0.0	22478.599609	23532.006358

So far, we have evaluated model performance by applying the k-fold cross validation technique on the train dataset. The `predict_model()` function makes predictions on the test dataset, thus letting us evaluate model performance on data that haven't been used to train the model. As we can see, the RMSE value on the test dataset is 4100.7982, indicating that the model has excellent performance.

[3] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html
[4] https://scikit-learn.org/

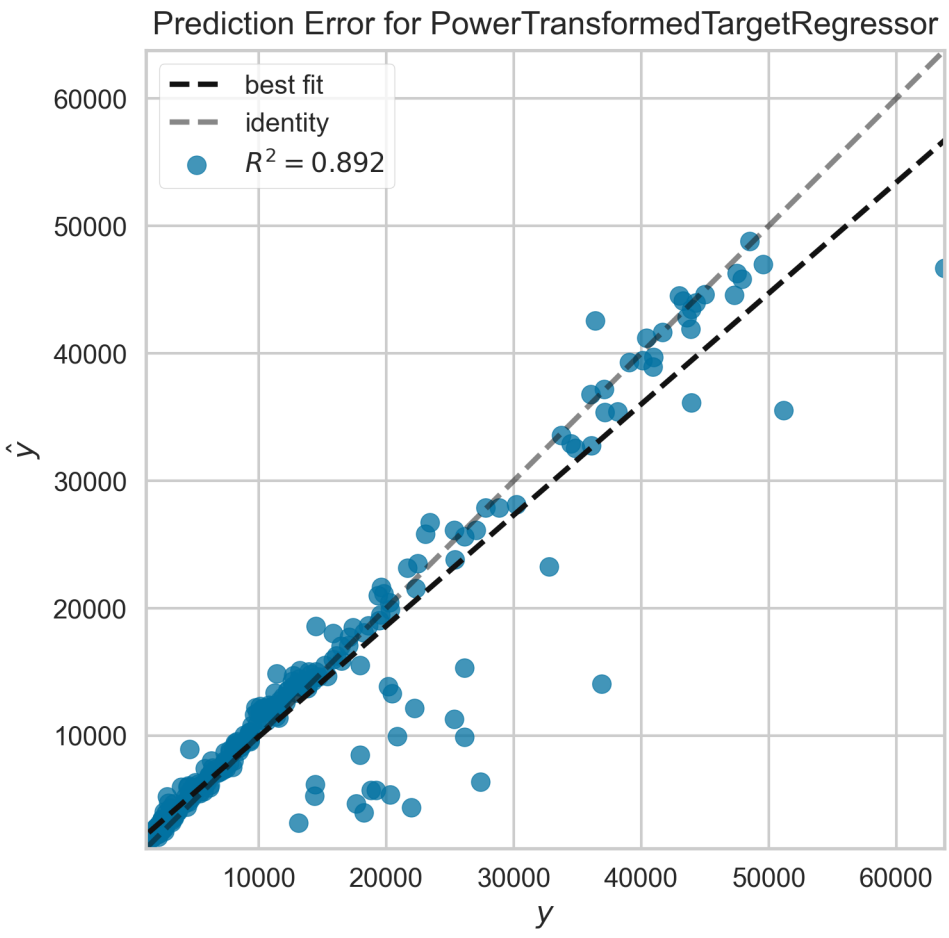
Plotting the Model

```
plot_model(tuned_model, 'feature', scale = 4)
```



The `plot_model()` function lets us easily create various useful graphs for our machine learning model. In this case, we created a feature importance plot to visualize the impact that each feature has at predicting the target variable. Evidently, the most important features are each person's age and being a smoker or not. This confirms the EDA insights, where we concluded that those features are indeed the most important. This function can also be used to plot the validation curve, the learning curve, and various other useful graphs about the regression model.

```
plot_model(tuned_model, 'error')
```



We now use the `plot_model()` function to create a prediction error plot, visualizing the difference between the dataset target values and the predictions of our model. As we can see, the model predictions are fairly accurate, because the best fit line is significantly close to the identity line, where all predictions are theoretically perfect. Furthermore, the R^2 metric also indicates that the model is accurate, having a value of 0.892.

Finalizing and Saving the Model

```
final_model = finalize_model(tuned_model)

save_model(final_model, 'regression_model')
```

As we saw earlier, the `setup()` function splits the dataset into train and test subsets. When we create or tune a model, only the train subset is used for training, while the rest of the data are reserved for testing purposes. In case we are satisfied with the performance of our model, we can use the `finalize_model()` function to train it on the complete dataset, thus utilizing the test subset as well. After doing that, we can use the `save_model()` function to save it on the local disk. We are going to see how this stored regression model can be deployed as a web application in a following chapter.

6. Classification

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Classification with PyCaret

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Importing the Necessary Libraries

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Loading the Dataset

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Exploratory Data Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Initializing the PyCaret Environment

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Identifying Numeric and Categorical Features

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Train/Test Split

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Normalization of Numeric Features

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Comparing Classification Models

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Creating the Model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Tuning the Model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Making Predictions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Plotting the Model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Finalizing and Saving the Model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

7. Clustering

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Clustering with PyCaret

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Importing the Necessary Libraries

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Generating a Synthetic Dataset

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Exploratory Data Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Initializing the PyCaret Environment

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Creating a Model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Plotting the Model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Saving and Assigning the Model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

8. Anomaly Detection

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Anomaly Detection with PyCaret

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Importing the Necessary Libraries

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Loading the Dataset

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Exploratory Data Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Initializing the PyCaret Environment

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Creating and Assigning the Model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Evaluating the Model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Plotting the Model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Saving the Model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

9. Deploying a Machine Learning Model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

The Streamlit Framework

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

The Insurance Charges Prediction App

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Developing the Web Application

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Running the Web Application Locally

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

The Iris Classification App

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Developing the Web Application

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Running the Web Application Locally

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Deploying an Application to Streamlit Cloud

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Creating a Github Repository

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Deploying the Insurance Charges Prediction App

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

10. Closing Thoughts

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pycaretbook.

Notes

A Brief Introduction to Machine Learning

1 Samuel, Arthur L. "Some studies in machine learning using the game of checkers." IBM Journal of research and development 3.3 (1959): 210-229.

2 Raschka, Sebastian. Python machine learning. Packt publishing ltd, 2015.

3 Russell, Stuart, and Peter Norvig. "Artificial intelligence: a modern approach." (2002).

4 Kaelbling, Leslie Pack, Michael L. Littman, and Andrew W. Moore. "Reinforcement learning: A survey." Journal of artificial intelligence research 4 (1996): 237-285.

5 Kiran, B. Ravi, et al. "Deep reinforcement learning for autonomous driving: A survey." IEEE Transactions on Intelligent Transportation Systems (2021).

6 Olive, David J. "Multiple linear regression." Linear regression. Springer, Cham, 2017. 17-83.

7 Tibshirani, Robert. "Regression shrinkage and selection via the lasso." Journal of the Royal Statistical Society: Series B (Methodological) 58.1 (1996): 267-288.

8 Ho, Tin Kam. "Random decision forests." Proceedings of 3rd international conference on document analysis and recognition. Vol. 1. IEEE, 1995.

9 Drucker, Harris, et al. "Support vector regression machines." Advances in neural information processing systems 9 (1997): 155-161.

10 Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." Annals of statistics (2001): 1189-1232.

11 Lantz, Brett. Machine learning with R: expert techniques for predictive modeling. Packt publishing ltd, 2019.