



Get started



Published in Towards Data Science · Follow



Moez Ali · Follow

Jun 18, 2020 · 11 min read



Build and deploy machine learning web app using PyCaret and Streamlit

A beginner's guide to deploying a machine learning app on Heroku PaaS



[Get started](#)

RECAP

In our [last post](#) on deploying a machine learning pipeline in the cloud, we demonstrated how to develop a machine learning pipeline in PyCaret, containerize Flask app with Docker and deploy serverless using AWS Fargate. If you haven't heard about PyCaret before, you can read this [announcement](#) to learn more.

In this tutorial, we will train a machine learning pipeline using PyCaret and





Get started

using a trained machine learning pipeline.

By the end of this tutorial, you will be able to build a fully functional web app to generate online predictions (one-by-one) and predictions by batch (by uploading a csv file) using trained machine learning model. The final app looks like this:

The screenshot shows a web browser window displaying the 'Insurance Charges Prediction App'. The app's interface includes a sidebar on the left with a 'Download you like to predict?' dropdown set to 'Online', a description 'This app is created to predict patient hospital charges', a link to 'https://www.pycaret.org', and a placeholder image of a modern building. The main content area features the 'PYCARET' logo and the title 'Insurance Charges Prediction App'. Below the title, there are input fields for 'Age' (25), 'Sex' (male), 'BMI' (19), 'Children' (0), a 'Smoker' checkbox, and a 'Region' dropdown (southwest). A 'Predict' button is located below these fields. At the bottom, a green box displays the output: 'The output is \$3546.3'. A red arrow points to the output box.

<https://pycaret-streamlit.herokuapp.com>





- What is a deployment and why do we deploy machine learning models?
- Develop a machine learning pipeline and train models using PyCaret.
- Build a simple web app using a Streamlit open-source framework.
- Deploy a web app on 'Heroku' and see the model in action.

This tutorial will cover the entire workflow starting from training a machine learning model and developing a pipeline in Python, developing a simple web app using streamlit and deploying the app on the Heroku cloud platform.

In the past, we have covered containerization using docker and deployment on cloud platforms like Azure, GCP and AWS. If you are interested in learning more about those, you can read the following stories:

- [Deploy Machine Learning Pipeline on AWS Fargate](#)
- [Deploy Machine Learning Pipeline on Google Kubernetes Engine](#)
- [Deploy Machine Learning Pipeline on AWS Web Service](#)
- [Build and deploy your first machine learning web app on Heroku PaaS](#)



[Get started](#)

PyCaret

PyCaret is an open source, low-code machine learning library in Python that is used to train and deploy machine learning pipelines and models into production. PyCaret can be installed easily using pip.

```
pip install pycaret
```

Streamlit

Streamlit is an open-source Python library that makes it easy to build beautiful custom web-apps for machine learning and data science. Streamlit can be installed easily using pip.

```
pip install streamlit
```



[Get started](#)

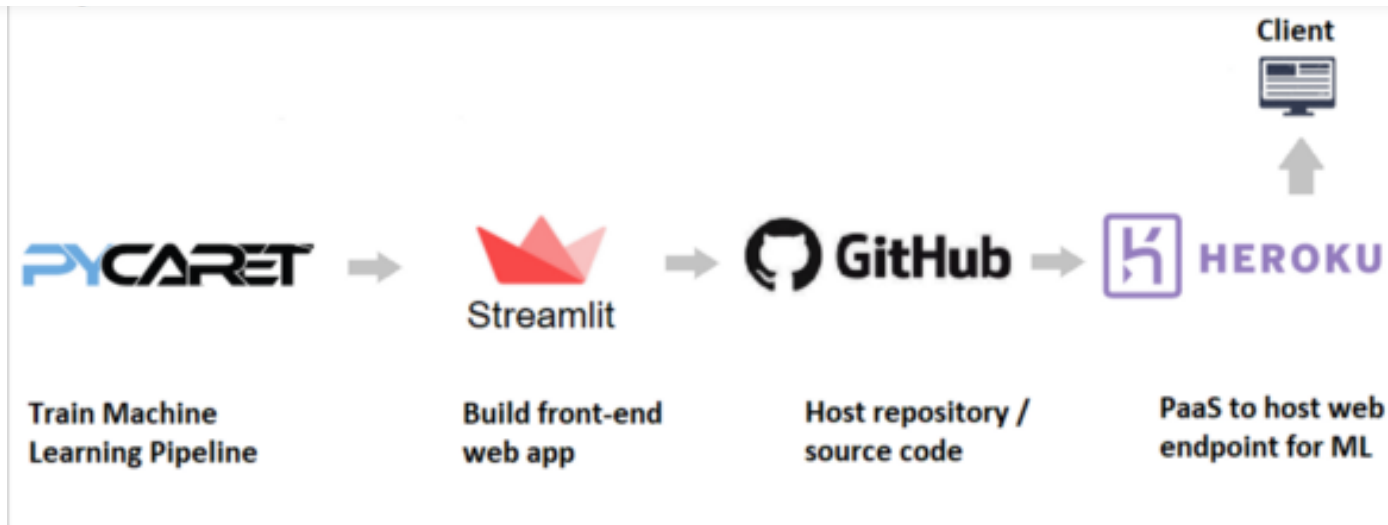
GitHub is a cloud-based service that is used to host, manage and control code.

Imagine you are working in a large team where multiple people (sometimes hundreds of them) are making changes. PyCaret is itself an example of an open-source project where hundreds of community developers are continuously contributing to source code. If you haven't used GitHub before, you can sign up for a free account.

Heroku

Heroku is a platform as a service (PaaS) that enables the deployment of web apps based on a managed container system, with integrated data services and a powerful ecosystem. In simple words, this will allow you to take the application from your local machine to the cloud so that anybody can access it using a Web URL. In this tutorial, we have chosen Heroku for deployment as it provides free resource hours when you sign up for a new account.



[Get started](#)

Machine Learning Workflow (from Training to Deployment on PaaS)

✓ **Let's get started.....**

Why Deploy Machine Learning Models?

Deployment of machine learning models is the process of putting models into production so that web applications, enterprise software and APIs can consume a trained model and generate predictions with new data points.





an outcome (binary value i.e. 1 or 0 for Classification, continuous values for Regression, labels for Clustering etc. There are two broad ways to predict new data points:

👉 Online Predictions

Online prediction scenarios are for cases where you want to generate predictions on a one-by-one basis for each datapoint. For example, you could use predictions to make immediate decisions about whether a particular transaction is likely to be fraudulent.

👉 Batch Predictions

Batch prediction is useful when you want to generate predictions for a set of observations all at once. For example, if you want to decide which customers to target as part of an advertisement campaign for a product you would get prediction scores for all customers, sort these to identify which customers are most likely to purchase, and then target maybe the top 5% customers that are most likely to purchase.





online prediction as well as batch prediction by uploading a csv file containing new data points.

Setting the Business Context

An insurance company wants to improve its cash flow forecasting by better predicting patient charges using demographic and basic patient health risk metrics at the time of hospitalization.

	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

(data source)





prediction using trained machine learning model and pipeline.

Tasks

- Train, validate and develop a machine learning pipeline using PyCaret.
- Build a front-end web application with two functionalities: (i) online prediction and (ii) batch prediction.
- Deploy the web app on Heroku. Once deployed, it will become publicly available and can be accessed via Web URL.

Task 1 — Model Training and Validation

Training and model validation are performed in an Integrated Development Environment (IDE) or Notebook either on your local machine or on cloud. If you haven't used PyCaret before, [click here](#) to learn more about PyCaret or see [Getting Started Tutorials](#) on our [website](#).

In this tutorial, we have performed two experiments. The first experiment is performed with default preprocessing settings in PyCaret. The second



[Get started](#)

into intervals. See the setup code for the second experiment:

Experiment No. 2

```
from pycaret.regression import *
```

```
r2 = setup(data, target = 'charges', session_id = 123,  
           normalize = True,  
           polynomial_features = True, trigonometry_features = True,  
           feature_interaction=True,  
           bin_numeric_features= ['age', 'bmi'])
```





	Description	Value
0	session_id	123
1	Transform Target	False
2	Transform Target Method	None
3	Original Data	(1338, 7)
4	Missing Values	False
5	Numeric Features	2
6	Categorical Features	4
7	Ordinal Features	False
8	High Cardinality Features	False
9	High Cardinality Method	None
10	Sampled Data	(1338, 7)
11	Transformed Train Set	(936, 14)
12	Transformed Test Set	(402, 14)

	Description	Value
0	session_id	123
1	Transform Target	False
2	Transform Target Method	None
3	Original Data	(1338, 7)
4	Missing Values	False
5	Numeric Features	2
6	Categorical Features	4
7	Ordinal Features	False
8	High Cardinality Features	False
9	High Cardinality Method	None
10	Sampled Data	(1338, 7)
11	Transformed Train Set	(936, 62)
12	Transformed Test Set	(402, 62)

Comparison of information grid for both experiments

The magic happens with only a few lines of code. Notice that in **Experiment 2** the transformed dataset has 62 features for training derived from only 6 features in the original dataset. All of the new features are the result of transformations and automatic feature engineering in PyCaret.

[Get started](#)

```
region_northeast', 'region_northwest', 'region_southeast',  
'region_southwest', 'age_0.0', 'age_1.0', 'age_10.0', 'age_11.0',  
'age_2.0', 'age_3.0', 'age_4.0', 'age_5.0', 'age_6.0', 'age_7.0',  
'bmi_8.0', 'bmi_9.0', 'smoker_yes_multiply_bmi_5.0',  
'region_southeast_multiply_age_Power2',  
'bmi_Power2_multiply_smoker_yes', 'age_Power2_multiply_smoker_yes',  
'children_0_multiply_bmi_Power2' ....
```

Columns in dataset after transformation

Sample code for model training in PyCaret:

```
# Model Training and Validation  
lr = create_model('lr')
```



[Get started](#)

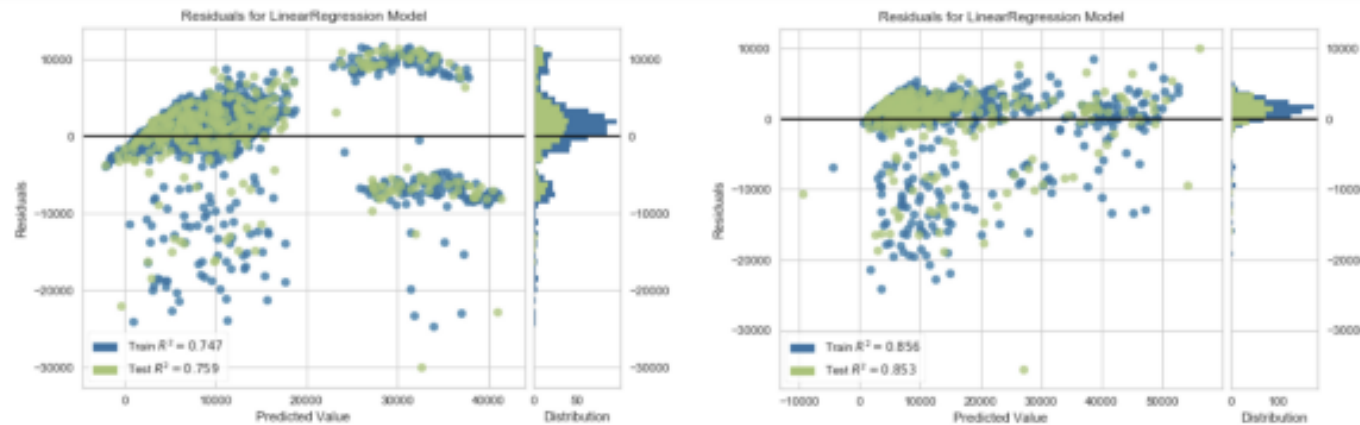
	MAE	MSE	RMSE	R2	RMSLE	MAPE		MAE	MSE	RMSE	R2	RMSLE	MAPE
0	4165.9659	3.330203e+07	5770.7909	0.8011	0.4683	0.4153	0	2424.4632	1.734011e+07	4164.1463	0.8964	0.3834	0.2802
1	4503.7366	4.374648e+07	6614.1122	0.7456	0.5633	0.4217	1	3407.0564	3.416396e+07	5844.9942	0.8014	0.4474	0.3340
2	3880.5528	3.179514e+07	5638.7179	0.5974	0.7645	0.4396	2	2928.2046	2.273657e+07	4768.2878	0.7121	0.5500	0.3787
3	3747.6457	2.680530e+07	5177.3833	0.7762	0.5015	0.5175	3	2926.7430	2.242633e+07	4735.6445	0.8127	0.5028	0.4021
4	4471.0419	4.341053e+07	6588.6670	0.6771	0.5224	0.3767	4	3101.1968	2.845052e+07	5333.9027	0.7884	0.5220	0.2826
5	4182.7551	3.616633e+07	6013.8450	0.7674	0.7416	0.4320	5	2975.2570	2.044595e+07	4521.7198	0.8685	0.3677	0.2883
6	4081.1022	3.919259e+07	6260.3984	0.7333	0.6434	0.4241	6	2720.7848	2.286434e+07	4781.6674	0.8444	0.4067	0.3331
7	4928.1534	4.641504e+07	6812.8581	0.7448	0.5887	0.4137	7	3124.1082	2.696739e+07	5193.0137	0.8517	0.4702	0.3092
8	4609.3147	4.037035e+07	6353.7670	0.7392	0.5686	0.5111	8	2820.0621	2.070707e+07	4550.5019	0.8663	0.3792	0.3189
9	4665.8647	4.259679e+07	6526.6220	0.7256	0.8131	0.4802	9	2985.9555	2.733254e+07	5228.0529	0.8239	0.4885	0.3361
Mean	4323.6133	3.838006e+07	6175.7162	0.7308	0.6175	0.4432	Mean	2941.3832	2.434348e+07	4912.1931	0.8266	0.4518	0.3263
SD	353.5472	5.908389e+06	490.4977	0.0543	0.1126	0.0431	SD	246.9597	4.637195e+06	462.4244	0.0495	0.0616	0.0381

10 Fold cross-validation of Linear Regression Model(s)

Notice the impact of transformations and automatic feature engineering. The R2 has increased by 10% with very little effort. We can compare the **residual plot** of linear regression model for both experiments and observe the impact of transformations and feature engineering on the **heteroskedasticity** of model.

```
# plot residuals of trained model
plot_model(lr, plot = 'residuals')
```





Residual Plot of Linear Regression Model(s)

Machine learning is an iterative process. The number of iterations and techniques used within are dependent on how critical the task is and what the impact will be if predictions are wrong. The severity and impact of a machine learning model to predict a patient outcome in real-time in the ICU of a hospital is far more than a model built to predict customer churn.

In this tutorial, we have performed only two iterations and the linear regression model from the second experiment will be used for deployment. At this stage, however, the model is still only an object within a Notebook / IDE. To save it as a



[Get started](#)

```
save_model(11, model_name = 'deployment_28042020',
```

When you save a model in PyCaret, the entire transformation pipeline based on the configuration defined in the **setup()** function is created. All inter-dependencies are orchestrated automatically. See the pipeline and model stored in the 'deployment_28042020' variable:





```
dataypes_Auto_Inter(categorical_features=[],
                    display_types=True, features_todrop=[],
                    ml_usecase='regression',
                    numerical_features=[], target='charges',
                    time_features=[])),
('imputer',
 Simple_Imputer(categorical_strategy='not_available',
                numeric_strategy='mean',
                target_variable=None)),
('new_levels1',
 New_Catagorical_Levels...
('dummy', Dummify(target='charges')),
('fix_perfect', Remove_100(target='charges')),
('clean_names', Clean_Colum_Names()),
('feature_select', Empty()), ('fix_multi', Empty()),
('dfs',
 DFS_Classic(interactions=['multiply'], ml_usecase='regression',
              random_state=123, subclass='binary',
              target='charges',
              top_features_to_pick_percentage=None)),
('pca', Empty())],
verbose=False),
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False),
None]
```

Pipeline created using PyCaret

We have finished training and model selection. The final machine learning pipeline and linear regression model is now saved as a pickle file (deployment_28042020.pkl) that will be used in a web application to generate



[Get started](#)

Now that our machine learning pipeline and model are ready we will start building a front-end web application that can generate predictions on new datapoints. This application will support 'Online' as well as 'Batch' predictions through a csv file upload. Let's breakdown the application code into three main parts:

Header / Layout

This section imports libraries, loads the trained model and creates a basic layout with a logo on top, a jpg image and a dropdown menu on the sidebar to toggle between 'Online' and 'Batch' prediction.



[Get started](#)

```
2 import streamlit as st
3 import pandas as pd
4 import numpy as np
5
6 model = load_model('deployment_28842820')
7
8 def predict(model, input_df):
9     predictions_df = predict_model(estimator=model, data=input_df)
10    predictions = predictions_df['label'][0]
11    return predictions
12
13 def run():
14
15     from PIL import Image
16     image = Image.open('logo.png')
17     image_hospital = Image.open('hospital.jpg')
18
19     st.image(image, use_column_width=False)
20
21     add_selectbox = st.sidebar.selectbox(
22         "How would you like to predict?",
23         ("Online", "Batch"))
24
25     st.sidebar.info('This app is created to predict patient hospital charges')
26     st.sidebar.success('https://www.pycaret.org')
27
28     st.sidebar.image(image_hospital)
29
30     st.title("Insurance Charges Prediction App")
```

Load trained model
for predictions

define function
to call

main function
starts here

adding sidebar

app.py — code snippet part 1

Online Predictions

This section deals with the first functionality of the app i.e. Online (one-by-one) prediction. We are using streamlit widgets such as *number input*, *text input*, *drop*





```
35 sex = st.selectbox('Sex', ['male', 'female'])
36 bmi = st.number_input('BMI', min_value=10, max_value=50, value=10)
37 children = st.selectbox('Children', [0,1,2,3,4,5,6,7,8,9,10])
38 if st.checkbox('Smoker'):
39     smoker = 'yes'
40 else:
41     smoker = 'no'
42 region = st.selectbox('Region', ['southwest', 'northwest', 'northeast', 'southeast'])
43
44 output=""
45
46 input_dict = {'age' : age, 'sex' : sex, 'bmi' : bmi, 'children' : children, 'smoker' : smoker, 'region' : region}
47 input_df = pd.DataFrame([input_dict])
48
49 if st.button("Predict"):
50     output = predict(model=model, input_df=input_df)
51     output = '$' + str(output)
52
53 st.success('The output is {}'.format(output))
```

required for
prediction using
streamlit widgets

calling predict function
when 'predict' button is
clicked

app.py — code snippet part 2

Batch Predictions

This part deals with the second functionality i.e. prediction by batch. We have used the **file_uploader** widget of streamlit to upload a csv file and then called the native **predict_model()** function from PyCaret to generate predictions that are displayed used streamlit's **write()** function.



[Get started](#)

```
58
59     if file_upload is not None:
60         data = pd.read_csv(file_upload)
61         predictions = predict_model(estimator=model, data=data)
62         st.write(predictions)
```

← generating data manager, the __predict__ widget and generate predictions using predict_model function of pycaret. This will return dataframe.

app.py — code snippet part 3

If you remember from Task 1 above we finalized a linear regression model that was trained on 62 features that were extracted using 6 original features. However, the front-end of our web application has an input form that collects only the six features i.e. age, sex, bmi, children, smoker, region.

How do we transform the 6 features of a new data point into 62 used to train the model? We do not need to worry about this part as PyCaret automatically handles this by orchestrating the transformation pipeline. When you call the predict function on a model trained using PyCaret, all transformations are applied automatically (in sequence) before generating predictions from the trained model.

Testing App

One final step before we publish the application on Heroku is to test the web app





Get started

localhost:5501


Apps WhatsApp Queens Email PyCaret Mail Wordpress HostPapa Github PyCaret PyPi Dify Jovian O YouTube Twitter AWS Management Microsoft Azure

How would you like to predict?

Online

This app is created to predict patient hospital charges

<https://www.pycaret.org>



pyCARET

Insurance Charges Prediction App

Age

25

Sex

male

BMI

50

Children

0

☐ Smoker

Region

southwest

Predict

The output is \$3646.0

Streamlit application testing — Online Prediction



[Get started](#)

How would you like to predict?

Batch

This app is created to predict patient hospital charges

<https://www.2vcareit.org>

PYCARET

Insurance Charges Prediction App

Upload our file for predictions

	age	sex	smr	children	smoke	region	charge
0	19	female	27.9900	0	yes	southwest	29990
1	18	male	22.7700	1	no	southwest	2650
2	26	male	31	0	no	southwest	3600
3	23	male	22.7000	0	no	northwest	4100
4	23	male	22.8000	0	no	northwest	3050
5	31	female	26.7000	0	no	southwest	4700
6	44	female	39.4000	1	no	southeast	19100
7	27	female	27.7000	0	no	northwest	6010
8	27	male	26.6000	2	no	northwest	18500
9	50	female	29.3000	0	no	northwest	14450
10	28	male	28.1000	0	no	northwest	8120

Streamlit application testing — Batch Prediction

👉 Task 3 — Deploy the Web App on Heroku

Now that the model is trained, the machine learning pipeline is ready, and the application is tested on our local machine, we are ready to start our deployment on Heroku. There are a couple of ways to upload your application source code onto Heroku. The simplest way is to link a GitHub repository to your Heroku account.

If you would like to follow along you can fork this [repository](#) from GitHub. If you don't know how to fork a repo, please read this [official GitHub tutorial](#)



[Get started](#)

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security 0 Insights Settings

PyCaret and Streamlit app deployment on Heroku

Edit

Manage topics

3 commits

1 branch

0 packages

0 releases

1 environment

1 contributor

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

pycaret updated setup.sh

Latest commit #d9694f 15 hours ago

Insurance - Model Training Notebook.ipynb

Add files via upload

15 hours ago

Procfile

Add files via upload

15 hours ago

README.md

Add files via upload

15 hours ago

app.py

Add files via upload

15 hours ago

deployment_28042020.pkl

Add files via upload

15 hours ago

hospital.jpg

Add files via upload

15 hours ago

insurance.csv

Add files via upload

15 hours ago

logo.png

Add files via upload

15 hours ago

requirements.txt

Add files via upload

15 hours ago

setup.sh

updated setup.sh

15 hours ago

<https://www.github.com/pycaret/pycaret-deployment-streamlit>

By now you are familiar with all of the files in the repository except for three files: 'requirements.txt', 'setup.sh' and 'Procfile'. Let's see what those are:



[Get started](#)

required to execute the application. If these packages are not installed in the environment where the application is running, it will fail.

2 lines (2 sloc) | 25 Bytes

Raw Blame History

```
1 pycaret==1.0.0
2 streamlit
```

setup.sh

setup.sh is a script programmed for bash. It contains instructions written in the Bash language and like requirements.txt, it is used for creating the necessary environment for our streamlit app to run on the cloud.

11 lines (11 sloc) | 219 Bytes

Raw Blame History

```
1 mkdir -p ~/.streamlit/
2 echo "\
3 [general]\n\
4 email = \"your-email@domain.com\"\n\
5 \" > ~/.streamlit/credentials.toml
6 echo "\
7 [server]\n\
8 headless = true\n\
9 enableCORS=false\n\
```



[Get started](#)

Procfile is simply one line of code that provides startup instructions to the web server that indicate which file should be executed when an application is triggered. In this example, 'Procfile' is used for executing **setup.sh** which will create the necessary environment for the streamlit app and the second part "streamlit run app.py" is to execute the application (this is similar to how you would execute a streamlit application on your local computer).

```
1 lines (1 sloc) | 48 Bytes
1 web: sh setup.sh && streamlit run app.py
```

Procfile

Once all the files are uploaded onto the GitHub repository, we are now ready to start deployment on Heroku. Follow the steps below:

Step 1 — Sign up on heroku.com and click on 'Create new app'





Get started



Welcome to Heroku

Now that your account has been set up, here's how to get started.

Dismiss



Create a new app

Create your first app and deploy your code to a running dyno.



Create new app



Create a team

Create teams to collaborate on your apps and pipelines.

Create a team

Looking for help getting started with your language?

Get started by reading one of our language guides in the Dev Center



Node.js



Ruby



Java



PHP



Python



Go



Scala



Clojure

Heroku Dashboard

Step 2 — Enter App name and region





Get started

App name

pycaret-streamlit



pycaret-streamlit is available

Choose a region



United States



Add to pipeline...

Create app



Heroku — Create new app

Step 3 — Connect to your GitHub repository





Get started

Add this app to a pipeline

Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features



Pipelines let you connect multiple apps together and **promote code** between them. [Learn more](#)



Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. [Learn more](#)

Choose a pipeline

Deployment method



Heroku Git
Use Heroku CLI



GitHub
Connect to GitHub



Container Registry
Use Heroku CLI

Connect to GitHub

Connect this app to GitHub to enable code diffs and deploys.

Search for a repository to connect to

pycaret

pycaret-deployment-streamlit

Search

Missing a GitHub organization? [Ensure Heroku Dashboard has team access](#)

Heroku — Connect to GitHub

Step 4 — Deploy branch





Get started

deployed to this app.

branch is always in a deployable state and any tests have passed before you push. [Learn more.](#)

Choose a branch to deploy

master

☐ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Enable Automatic Deploys

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more.](#)

Choose a branch to deploy

master

Deploy Branch



Heroku — Deploy Branch

Step 5 — Wait 10 minutes and BOOM

App is published to URL: <https://pycaret-streamlit.herokuapp.com/>





Get started

How would you like to predict?

Online

This app is created to predict patient hospital charges

<https://www.pycaret.org>

PYCARET

Insurance Charges Prediction App

Age: 25

Sex: male

BMS: 10

Children: 0

☐ Smoker

Region: southwest

Predict

The output is \$2546.0

<https://pycaret-streamlit.herokuapp.com/>

PyCaret 2.0.0 is coming!

We have received overwhelming support and feedback from the community. We are actively working on improving PyCaret and preparing for our next release.

PyCaret 2.0.0 will be bigger and better. If you would like to share your feedback and help us improve further, you may fill this form on the website or



[Get started](#)

PyCaret.

Want to learn about a specific module?

As of the first release 1.0.0, PyCaret has the following modules available for use. Click on the links below to see the documentation and working examples in Python.

[Classification](#)

[Regression](#)

[Clustering](#)

[Anomaly Detection](#)

[Natural Language Processing](#)

[Association Rule Mining](#)

Also see:

PyCaret getting started tutorials in Notebook:



[Get started](#)

[ASSOCIATION RULE MINING](#)

[Regression](#)

[Classification](#)

Would you like to contribute?

PyCaret is an open source project. Everybody is welcome to contribute. If you would like to contribute, please feel free to work on [open issues](#). Pull requests are accepted with unit tests on dev-1.0.1 branch.

Please give us  on our [GitHub repo](#) if you like PyCaret.

Medium: https://medium.com/@moez_62905/

LinkedIn: <https://www.linkedin.com/in/profile-moez/>

Twitter: <https://twitter.com/moezpycaretorg1>





Get started

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. [Take a look.](#)

Get this newsletter

