Nutan Follow

Dec 11, 2020 · 3 min read · Listen

Deploy Machine Learning Model in Google Cloud Platform Using Flask — Part 1

Our goal is to predict sales on the basis of the three media budgets TV, Radio and Newspaper.

Machine Learning Model Deployment

Dataset Information

The Advertising dataset consists of the sales of that product in 200 different markets, along with advertising budgets for the product in each of those markets for three different media: TV, radio and newspaper.

We try to determine that there is an association between advertising and sales, so that we can instruct our clients to adjust advertising budgets, thereby indirectly increasing sales.

This dataset shared by Marshall School of Business at the University of Southern California. If you want, you can download

Load Data

```
import pandas as pd

df = pd.read_csv('input/adver
df.head()
```

Output:

	Unnamed: 0	TV	radio	newspaper	sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

```
df.columns
```

Output:

```
Index(['Unnamed: 0', 'TV', 'radio', 'newspaper', 'sales'], dtype='object')
```

```
df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  200 non-null    int64
 1   TV          200 non-null    float64
 2   radio       200 non-null    float64
 3   newspaper   200 non-null    float64
 4   sales       200 non-null    float64
dtypes: float64(4), int64(1)
memory usage: 7.9 KB
```

Drop unnamed column

```
df = df.drop(['Unnamed: 0'], axis=1)
df.head()
```

Output:

	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

Split data into feature and target

```
X = df.loc[:, df.columns != 'sales']
X.head()
```

Output:

	TV	radio	newspaper
0	230.1	37.8	69.2
1	44.5	39.3	45.1
2	17.2	45.9	69.3
3	151.5	41.3	58.5
4	180.8	10.8	58.4

```
y = df['sales']
y.head()
```

Output:

```
0    22.1
1    10.4
2     9.3
3    18.5
4    12.9
Name: sales, dtype: float64
```

Split data into train and test

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,
y,test_size=0.25)

print(X_train.shape)
print(X_train.head())
```

Output:

```
(150, 3)
        TV  radio  newspaper
155    4.1   11.6        5.7
174  222.4    3.4       13.1
184  253.8   21.3       30.0
23   228.3   16.9       26.2
133  219.8   33.5       45.1
```

```
print(X_test.shape)
print(X_test.head())
```

Output:

```
(50, 3)
       TV  radio  newspaper
122  224.0    2.4       15.6
186  139.5    2.1       26.6
5      8.7   48.9       75.0
55   198.9   49.4       60.0
194  149.7   35.6        6.0
```

```
print(y_train.shape)
print(y_train.head())
```

Output:

```
(150,)
155     3.2
174    11.5
184    17.6
23     15.5
133    19.6
Name: sales, dtype: float64
```

```
print(y_test.shape)
print(y_test.head())
```

Output:

```
(50,)
122    11.6
186    10.3
5       7.2
55     23.7
194    17.3
Name: sales, dtype: float64
```

Model creation

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

Training

```
model.fit(X_train, y_train)
```

Predict

```
predictions = model.predict(X_test)
predictions[:10]
```

Output:

array([13.63521962, 9.78796421, 12.43510316, 21.10986043, 16.14591182, 22.97996462, 12.24553359, 10.14917755, 15.21272804, 18.44335937])

Save the model

joblib module

We are using joblib library to save our model.

joblib.dump() and joblib.load() provide a replacement for pickle to work efficiently on arbitrary Python objects containing large data, in particular large numpy arrays.

joblib.dump(value, filename, compress=0, protocol=None, cache_size=None): Persist an arbitrary Python object into one file.

```
import joblib
joblib.dump(model,'output/lr_model.pkl')
```

Predict with new data

```
import numpy as np
test_data = [159.1, 60.2, 90]
#Convert into numpy array and reshape
test_data = np.array(test_data).astype(np.float)
test_data = test_data.reshape(1,-1)
test_data
```

Load saved model

```
filePath = 'output/lr_model.pkl'
file = open(filePath, "rb")
trained_model = joblib.load(file)
```

Predict

```
prediction = trained_model.predict(test_data)
print(prediction)
```

Output:

[21.37257034]

Ml Model Deployment Model Deployment Flask Ml Deployment On Gcp

Google Cloud Platform Ml Flask Deployment