Data Wrangling and Analysis: A Python Journey with Car Crashes, Titanic, and Tips

Mustafa Germec, PhD · Follow
Published in Dev Genius · 11 min read · 6 days ago

1

Photo by Claudio Schwarz on Unsplash

Abstract

This paper describes a series of Python code snippets and analyses performed on two datasets, ‘car_crashes’, ‘titanic’, and ‘tips’, utilizing libraries such as NumPy, Pandas, Seaborn, and Matplotlib.

The first part involves data preprocessing on the 'car_crashes' dataset, where list comprehension is employed to modify variable names based on data type and presence of specific substrings. Additionally, a new dataframe is created by selecting variables not present in a given list.

The 'titanic' and 'tips' datasets are then analyzed extensively. Operations include data exploration, visualization, and manipulation. Techniques such as grouping, filtering, filling missing values, creating new variables, and sorting are utilized to extract insights and perform transformations. For instance, the code segments find counts of passengers based on various criteria, calculate statistics like means and sums, and generate visualizations. Finally, functions are defined and applied to derive additional variables based on specified conditions.

Overall, these code snippets showcase a comprehensive data analysis pipeline, covering data preprocessing, exploration, and transformation techniques applied to real-world datasets.

1. Importing the libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
```

```
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 500)
```

2. Let's use the List Comprehension structure to capitalize the names of the numeric variables in the car_crashes data and add NUM to the beginning. Note: Non-numeric names should also grow. Must be done with a single list comprehension structure.

```
car_crashes = sns.load_dataset('car_crashes')
car_crashes.head()
print(['NUM_' + col.upper() if car_crashes[col].dtype != 'O' else col.upper() fo
```

```
['NUM_TOTAL', 'NUM_SPEEDING', 'NUM_ALCOHOL', 'NUM_NOT_DISTRACTED', 'NUM_NO_PREVI
```

3. Using the List Comprehension structure, write "FLAG" after the names of the variables that do not contain "no" in the car_crashes data. Note: All variable names must be uppercase. A single list should be made with comprehension.

```
) + '_FLAG' if 'no' not in col else col.upper() for col in car_crashes.columns])
```

```
['TOTAL_FLAG', 'SPEEDING_FLAG', 'ALCOHOL_FLAG', 'NOT_DISTRACTED', 'NO_PREVIOUS',
```

4. Let's choose the names of the variables that are DIFFERENT from the variable names given below using the List Comprehension structure and create a new dataframe. Note: First, let's create a new list named new_cols using list comprehension according to the list above. Then let's create a new df by selecting these variables with df[new_cols] and name it new_df

```
given_list = ['abbrev', 'no_previous']

new_list = [col for col in car_crashes.columns if col not in given_list]

new_df = car_crashes[new_list]

new_df.head()
```

```
   total  speeding  alcohol  not_distracted  ins_premium  ins_losses
0   18.8     7.332    5.640          18.048       784.55      145.08
1   18.1     7.421    4.525          16.290      1053.48      133.93
2   18.6     6.510    5.208          15.624       899.47      110.35
```

```
3   22.4     4.032    5.824          21.056       827.34      142.39
4   12.0     4.200    3.360          10.920       878.41      165.63
```

5. Let’s define the Titanic dataset from the Seaborn library

```
titanic = sns.load_dataset('titanic')
titanic.head()
```

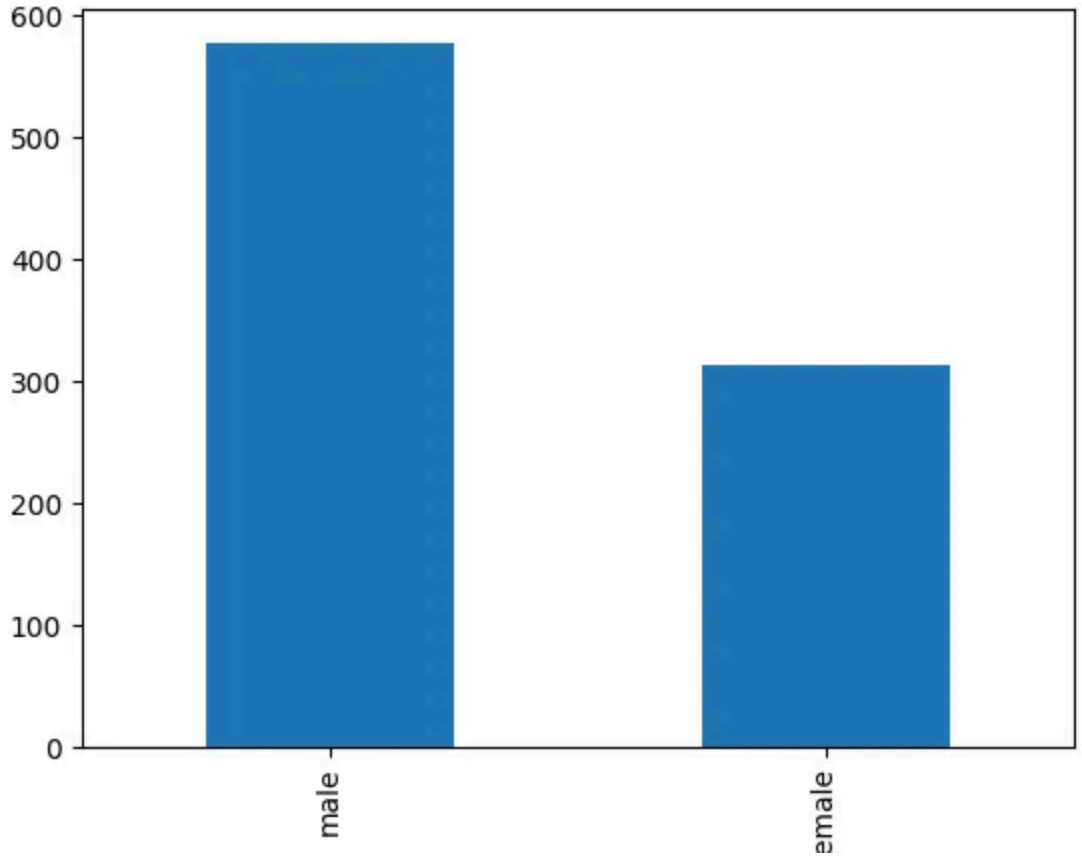
```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class    who
0         0       3    male  22.0      1      0   7.2500        S  Third    man
1         1       1  female  38.0      1      0  71.2833        C  First  woman
2         1       3  female  26.0      0      0   7.9250        S  Third  woman
3         1       1  female  35.0      1      0  53.1000        S  First  woman
4         0       3    male  35.0      0      0   8.0500        S  Third    man
```

6: Let’s find the number of female and target passengers in the Titanic dataset

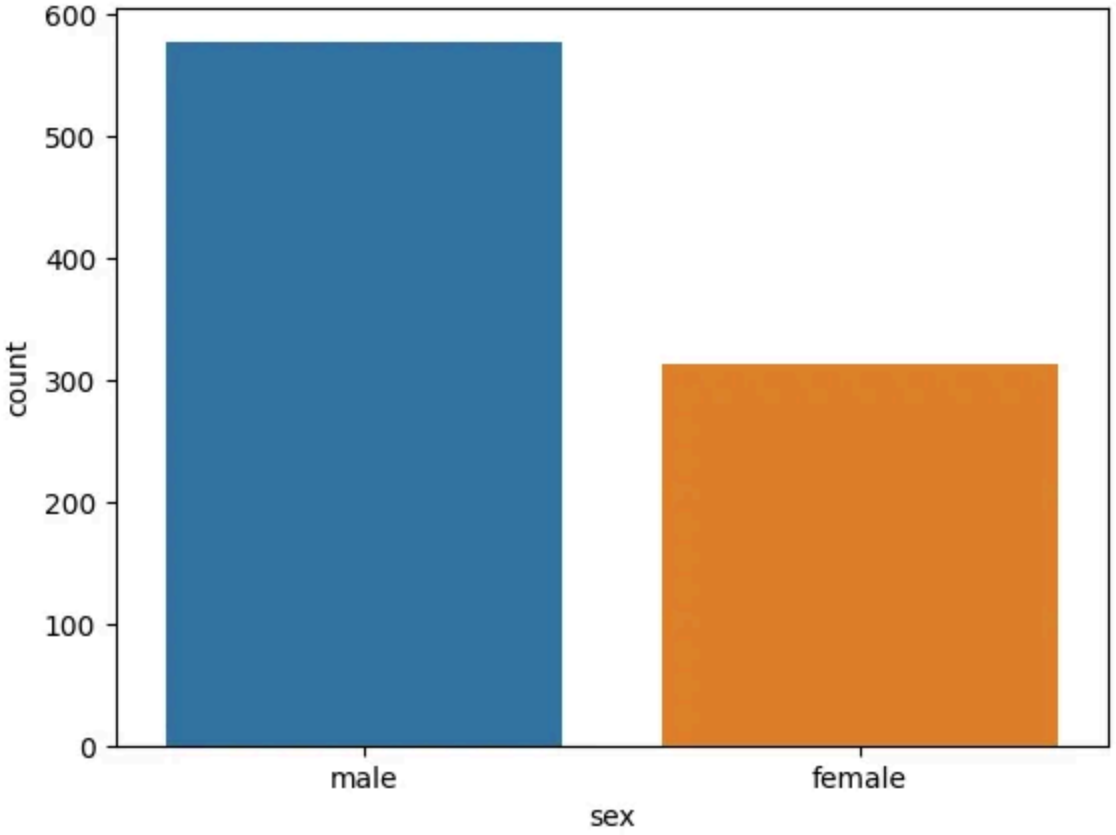
```
titanic['sex'].value_counts()
```

```
male      577
female    314
Name: sex, dtype: int64
```

```
titanic.sex.value_counts().plot(kind='bar')
plt.show()
```




```
sns.countplot(titanic, x='sex')
plt.show()
```



7. Let's find the number of unique values for each column

```
titanic.nunique().sort_values(ascending=False)
```

```
fare           248
age             88
sibsp            7
parch            7
deck             7
pclass           3
embarked         3
class            3
who              3
embark_town      3
survived         2
sex              2
adult_male       2
alive            2
alone            2
dtype: int64
```

8. Let’s find the unique values of the pclass variable

```
titanic['pclass'].unique()
```

```
array([3, 1, 2], dtype=int64)
```

9. Let’s find the number of unique values of pclass and parch variables

```
variables = ['pclass', 'parch']
titanic[variables].nunique()
```

```
pclass    3
parch     7
dtype: int64
```

10. Let’s check the type of the embarked variable. Let’s change its type to category. Let’s check the type again.

```
titanic['embarked'].dtype
titanic['embarked'] = titanic['embarked'].astype('category')
titanic['embarked'].dtype
```

```
dtype('O')
CategoricalDtype(categories=['C', 'Q', 'S'], ordered=False)
```

11. Let’s show all the sages of embarked value C.

```
titanic[titanic['embarked'] == 'C'].head()
```

```
     survived  pclass     sex   age  sibsp  parch     fare embarked   class    wh
1           1       1  female  38.0      1      0  71.2833        C   First  woma
9           1       2  female  14.0      1      0  30.0708        C  Second  chil
19          1       3  female   NaN      0      0   7.2250        C   Third  woma
26          0       3    male   NaN      0      0   7.2250        C   Third    ma
30          0       1    male  40.0      0      0  27.7208        C   First    ma
```

12. Let’s show all the sages of those with no embarked value S.

```
titanic[titanic['embarked'] != 'S'].head()
```

```
     survived  pclass     sex   age  sibsp  parch     fare embarked   class    wh
1           1       1  female  38.0      1      0  71.2833        C   First  woma
5           0       3    male   NaN      0      0   8.4583        Q   Third    ma
9           1       2  female  14.0      1      0  30.0708        C  Second  chil
16          0       3    male   2.0      4      1  29.1250        Q   Third  chil
19          1       3  female   NaN      0      0   7.2250        C   Third  woma
```

13. Let's show all the information of passengers younger than 30 and female

```
titanic[(titanic['age'] < 30) & (titanic['sex'] == 'female')].head()
```

```
    survived  pclass     sex   age  sibsp  parch     fare embarked   class    wh
2          1       3  female  26.0      0      0   7.9250        S   Third  woma
8          1       3  female  27.0      0      2  11.1333        S   Third  woma
9          1       2  female  14.0      1      0  30.0708        C  Second  chil
10         1       3  female   4.0      1      1  16.7000        S   Third  chil
14         0       3  female  14.0      0      0   7.8542        S   Third  chil
```

14. Let's show fare information for passengers over 500 fare or 70 years old

```
titanic[(titanic['fare'] > 500) | (titanic['age'] > 70)]
```

```
     survived  pclass     sex   age  sibsp  parch      fare embarked  class    w
96          0       1    male  71.0      0      0   34.6542        C  First    m
116         0       3    male  70.5      0      0    7.7500        Q  Third    m
258         1       1  female  35.0      0      0  512.3292        C  First  wom
```

```
493         0       1    male  71.0      0      0   49.5042        C  First    m
630         1       1    male  80.0      0      0   30.0000        S  First    m
679         1       1    male  36.0      0      1  512.3292        C  First    m
737         1       1    male  35.0      0      0  512.3292        C  First    m
851         0       3    male  74.0      0      0    7.7750        S  Third    m
```

15. Let’s find the sum of the null values in each variable

```
titanic.isna().sum().sort_values(ascending=False)
```

```
deck           688
age            177
embarked         2
embark_town      2
survived         0
pclass           0
sex              0
sibsp            0
parch            0
fare             0
class            0
who              0
adult_male       0
alive            0
alone            0
dtype: int64
```

16. Let's drop the variable "who" from the dataframe

```
titanic.drop('who', axis=1, inplace=True)
titanic.head()
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  adult_
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First       F
2         1       3  female  26.0      0      0   7.9250        S  Third       F
3         1       1  female  35.0      1      0  53.1000        S  First       F
4         0       3    male  35.0      0      0   8.0500        S  Third
```

17. Let's fill the empty values in the "deck" variable with the most repeated value (mode) of the deck variable

```
most_repeated_value = titanic['deck'].mode()  # 0    C
titanic['deck'].fillna(most_repeated_value[0])
```



```
1      C
2      C
3      C
```

```
4      C
      ..
886    C
887    B
888    C
889    C
890    C
Name: deck, Length: 891, dtype: category
Categories (7, object): ['A', 'B', 'C', 'D', 'E', 'F', 'G']
```

18. Let's fill the empty values in the "age" variable with the median of the age variable

```
median_value_of_age = titanic['age'].median()   # 28.0
titanic['age'].fillna(median_value_of_age)
```

```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    28.0
889    26.0
```

```
890    32.0
Name: age, Length: 891, dtype: float64
```

19. Let's find the sum, count, mean values of the survived variable by pclass and sex variables

```
titanic.groupby(['pclass', 'sex']).agg({'survived': ['sum', 'count', 'mean']})
```

```
              survived
                   sum count      mean
pclass sex
1      female       91    94  0.968085
       male         45   122  0.368852
2      female       70    76  0.921053
       male         17   108  0.157407
3      female       72   144  0.500000
       male         47   347  0.135447
```

20. Let's write a function that returns 1 for those under 30 and 0 for those above or equal to 30. Let's create a variable named age_flag in the titanic data set using the written function. (let's use apply and lambda constructs)

First solution

```
def age_30(age):
    if age > 30:
        return 1
    else:
        return 0

titanic['age_flag_1'] = titanic['age'].apply(lambda age: age_30(age))
titanic.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who
0	0	3	male	22.0	1	0	7.2500	S	Third	man
1	1	1	female	38.0	1	0	71.2833	C	First	woman
2	1	3	female	26.0	0	0	7.9250	S	Third	woman
3	1	1	female	35.0	1	0	53.1000	S	First	woman
4	0	3	male	35.0	0	0	8.0500	S	Third	man

Second solution

```
titanic['age_flag_2'] = titanic['age'].apply(lambda age: 1 if age > 30 else 0)
titanic.head()
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class    who
0         0       3    male  22.0      1      0   7.2500        S  Third    man
1         1       1  female  38.0      1      0  71.2833        C  First  woman
2         1       3  female  26.0      0      0   7.9250        S  Third  woman
3         1       1  female  35.0      1      0  53.1000        S  First  woman
4         0       3    male  35.0      0      0   8.0500        S  Third    man
```

Third solution

```
def age_variable(dataframe, variable):
    dataframe['age_flag_3'] = [1 if variable > 30 else 0 for variable in datafra

age_variable(titanic, 'age')
titanic.head()
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked  class    who
0         0       3    male  22.0      1      0   7.2500        S  Third    man
1         1       1  female  38.0      1      0  71.2833        C  First  woman
2         1       3  female  26.0      0      0   7.9250        S  Third  woman
3         1       1  female  35.0      1      0  53.1000        S  First  woman
4         0       3    male  35.0      0      0   8.0500        S  Third    man
```

21. Let’s define the Tips dataset from the Seaborn library

```
tips = sns.load_dataset('tips')
tips.head()
```

```
   total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4
```

22. Let’s find the sum, min, max and average of total_bill values according to the categories (Dinner, Lunch) of the time variable

```
tips.groupby('time').agg({'total_bill': ['sum', 'min', 'max', 'mean']})
```

```
       total_bill
              sum   min    max       mean
time
```

```
Lunch     1167.47  7.51  43.11  17.168676
Dinner    3660.30  3.07  50.81  20.797159
```

23. Let’s find the sum, min, max and average of total_bill values by days and time

```
tips.groupby(['day', 'time']).agg({'total_bill': ['sum', 'min', 'max', 'mean']})
```

```
            total_bill
                   sum    min    max       mean
day  time
Thur Lunch     1077.55   7.51  43.11  17.664754
     Dinner      18.78  18.78  18.78  18.780000
Fri  Lunch       89.92   8.58  16.27  12.845714
     Dinner     235.96   5.75  40.17  19.663333
Sat  Lunch        0.00    NaN    NaN        NaN
     Dinner    1778.40   3.07  50.81  20.441379
Sun  Lunch        0.00    NaN    NaN        NaN
     Dinner    1627.16   7.25  48.17  21.410000
```

24. Let’s find the sum, min, max and average of the total_bill and tip values of the lunchtime and female customers according to the day.

```
tips[(tips['time'] == 'Lunch') & (tips['sex']=='Female')].groupby('day').agg({
    'total_bill': ['sum', 'min', 'max', 'mean'],
    'tip': ['sum', 'min', 'max', 'mean']
})
```

```
     total_bill                              tip
            sum    min    max      mean    sum   min   max      mean
day
Thur     516.11   8.35  43.11  16.64871  79.42  1.25  5.17  2.561935
Fri       55.76  10.09  16.27  13.94000  10.98  2.00  3.48  2.745000
Sat        0.00    NaN    NaN       NaN   0.00   NaN   NaN       NaN
Sun        0.00    NaN    NaN       NaN   0.00   NaN   NaN       NaN
```

25. Let’s find the average of orders with size less than 3 and total_bill greater than 10

```
tips.loc[(tips['size'] < 3) & (tips['total_bill'] > 10), 'total_bill'].mean()
```

```
17.184965034965035
```

26. Let's create a new variable called total_bill_tip_sum. Return the sum of the total_bill and tip each customer paid.

```
tips['total_bill_tip_sum'] = tips['total_bill'] + tips['tip']
tips.head()
```

```
   total_bill   tip     sex smoker  day    time  size  total_bill_tip_sum
0       16.99  1.01  Female     No  Sun  Dinner     2               18.00
1       10.34  1.66    Male     No  Sun  Dinner     3               12.00
2       21.01  3.50    Male     No  Sun  Dinner     3               24.51
3       23.68  3.31    Male     No  Sun  Dinner     2               26.99
4       24.59  3.61  Female     No  Sun  Dinner     4               28.20
```

27. Let's sort the total_bill type_sum variable from largest to smallest and assign the first 30 people to a new dataframe

```
new_tips = tips.sort_values('total_bill_tip_sum', ascending=False).head(30)
new_tips
```

```
     total_bill    tip     sex smoker   day    time  size  total_bill_tip_sum
170       50.81  10.00    Male    Yes   Sat  Dinner     3               60.81
212       48.33   9.00    Male     No   Sat  Dinner     4               57.33
```

```
59        48.27   6.73    Male     No   Sat  Dinner     4                55.00
156       48.17   5.00    Male     No   Sun  Dinner     6                53.17
182       45.35   3.50    Male    Yes   Sun  Dinner     3                48.85
197       43.11   5.00  Female    Yes  Thur   Lunch     4                48.11
23        39.42   7.58    Male     No   Sat  Dinner     4                47.00
102       44.30   2.50  Female    Yes   Sat  Dinner     3                46.80
142       41.19   5.00    Male     No  Thur   Lunch     5                46.19
95        40.17   4.73    Male    Yes   Fri  Dinner     4                44.90
184       40.55   3.00    Male    Yes   Sun  Dinner     2                43.55
112       38.07   4.00    Male     No   Sun  Dinner     3                42.07
207       38.73   3.00    Male    Yes   Sat  Dinner     4                41.73
56        38.01   3.00    Male    Yes   Sat  Dinner     4                41.01
141       34.30   6.70    Male     No  Thur   Lunch     6                41.00
238       35.83   4.67  Female     No   Sat  Dinner     3                40.50
11        35.26   5.00  Female     No   Sun  Dinner     4                40.26
52        34.81   5.20  Female     No   Sun  Dinner     4                40.01
85        34.83   5.17  Female     No  Thur   Lunch     4                40.00
47        32.40   6.00    Male     No   Sun  Dinner     4                38.40
180       34.65   3.68    Male    Yes   Sun  Dinner     4                38.33
179       34.63   3.55    Male    Yes   Sun  Dinner     2                38.18
83        32.68   5.00    Male    Yes  Thur   Lunch     2                37.68
39        31.27   5.00    Male     No   Sat  Dinner     3                36.27
167       31.71   4.50    Male     No   Sun  Dinner     4                36.21
175       32.90   3.11    Male    Yes   Sun  Dinner     2                36.01
44        30.40   5.60    Male     No   Sun  Dinner     4                36.00
173       31.85   3.18    Male    Yes   Sun  Dinner     2                35.03
116       29.93   5.07    Male     No   Sun  Dinner     4                35.00
155       29.85   5.14  Female     No   Sun  Dinner     5                34.99
```

Conclusions

In conclusion, the provided series of data manipulation tasks involving Python's pandas and seaborn libraries demonstrate diverse techniques for

data exploration, cleaning, and analysis. Beginning with data loading and exploration, the tasks progress through various operations including data filtering, variable renaming, grouping, aggregation, and creating derived variables. Techniques such as list comprehensions, lambda functions, and method chaining are employed efficiently to perform these operations succinctly. The tasks cover a wide range of data analysis scenarios including handling missing values, creating new variables based on conditions, grouping data, and generating summary statistics. Overall, these exercises illustrate fundamental data manipulation techniques essential for exploratory data analysis and modeling.

Data Analysis | Data Wrangling | Data Science | Python | Machine Learning

Written by Mustafa Germec, PhD

Follow

306 Followers · Writer for Dev Genius

I am interested in bioprocessing, data science, machine learning, natural language process (NLP), time series, and structured query language (SQL).

More from Mustafa Germec, PhD and Dev Genius

Mustafa Germec, PhD

Exploring NYC Public School Test Result Scores: A DataCamp Project

Every year, American high school students take SATs, which are standardized tests...

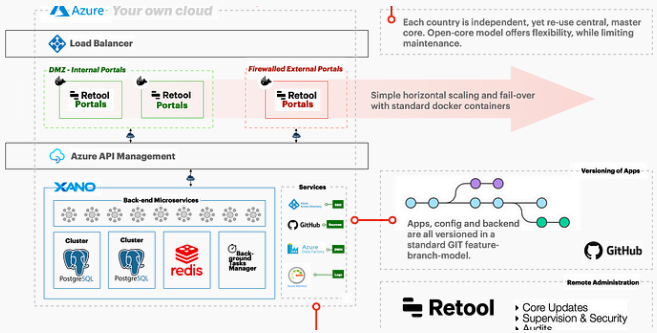
Anusha SP in Dev Genius

Java 8 Coding and Programming Interview Questions and Answers

It has been 8 years since Java 8 was released. I have already shared the Java 8 Interview...

3 min read · Feb 25, 2024

60



Maxime Topolov in Dev Genius

Low-code for large projects. It's time.

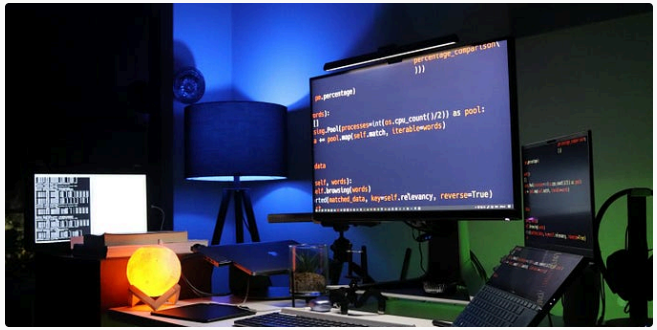
After reading this article, you'll be convinced to use low-code for your company's next...

· 8 min read · Feb 14, 2024

688 19

6 min read · Jan 31, 2023

668 9



Mustafa Germec, PhD in AI Mind

Optimizing LightGBM Classifier Hyperparameters with Optuna

Abstract

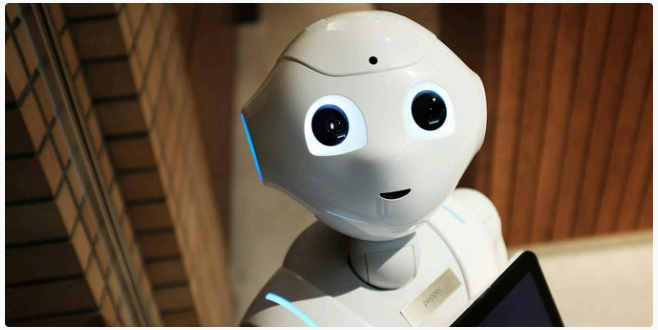
11 min read · Nov 9, 2023

137

See all from Mustafa Germec, PhD

See all from Dev Genius

Recommended from Medium



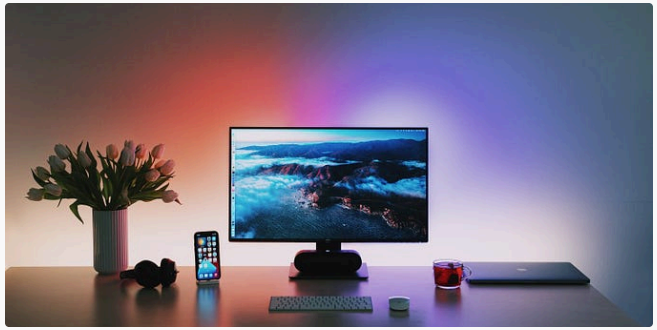
Abhinaba Banerjee in Python in Plain English

End-to-end Deep Learning Project with AWS Deployment (Part 1)

The main objective of the project is to identify kidney images that are tumor or normal. The...

8 min read · 5 days ago

15



Arunn Thevapalan in Towards Data Science

Building Your First Desktop Application using PySide6 [A Dat...

Surprise, surprise. It's not as hard as I thought it would be.

· 14 min read · Mar 16, 2024

975 6

Lists