ML basics: Loan prediction

The complete Data Science pipeline on a simple problem

Tariq Massaoudi · Follow
Published in Towards Data Science · 7 min read · Jun 6, 2019

Photo by Dmitry Demidko on Unsplash

The problem:

Dream Housing Finance company deals in all home loans. They have presence across all urban, semi urban and rural areas. Customer first apply for home loan after that company validates the customer eligibility for loan.

The Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have given a problem to identify the customers segments, those are eligible for loan amount so that they can specifically target these customers.

It's a classification problem , given information about the application we have to predict whether the they'll be to pay the loan or not.

We'll start by exploratory data analysis , then preprocessing , and finally we'll be testing different models such as Logistic regression and decision trees.

The data consists of the following rows:

```
Loan_ID : Unique Loan ID
Gender : Male/ Female
Married : Applicant married (Y/N)
Dependents : Number of dependents
Education : Applicant Education (Graduate/ Under Graduate)
Self_Employed : Self employed (Y/N)
ApplicantIncome : Applicant income
CoapplicantIncome : Coapplicant income
LoanAmount : Loan amount in thousands of dollars
Loan_Amount_Term : Term of loan in months
Credit_History : credit history meets guidelines yes or no
Property_Area : Urban/ Semi Urban/ Rural
Loan_Status : Loan approved (Y/N) this is the target variable
```

Exploratory data analysis:

We'll be using seaborn for visualisation and pandas for data manipulation. You can download the dataset from here : https://datahack.analyticsvidhya.com/contest/practice-problem-loan-prediction-iii/

We'll import the necessary libraries and load the data :

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
import numpy as np

train=pd.read_csv("train.csv")
test=pd.read_csv("test.csv")
```

We can look at few top rows using the head function

```
train.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Pro
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	

Image by Author

We can see that there's some missing data , we can further explore this using the pandas describe function:

```
train.describe()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

Image by Author

Some variables have missing values that we'll have to deal with , and also there seems to be some outliers for the Applicant Income , Coapplicant income and Loan Amount . We also see that about 84% applicants have a credit_history. Because the mean of Credit_History field is 0.84 and it has either (1 for having a credit history or 0 for not)

It would be interesting to study the distribution of the numerical variables mainly the Applicant income and the loan amount. To do this we'll use seaborn for visualization.

```
sns.distplot(train.ApplicantIncome,kde=False)
```

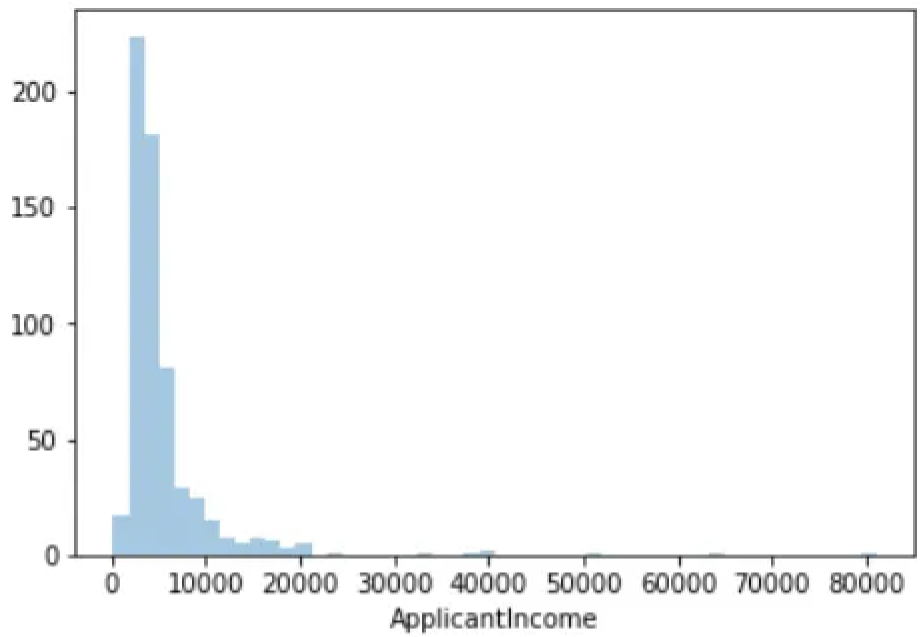


Image by Author

The distribution is skewed and we can notice quite a few outliers.

Since Loan Amount has missing values , we can't plot it directly. One solution is to drop the missing values rows then plot it, we can do this using the dropna function

```
sns.distplot(train.ApplicantIncome.dropna(),kde=False)
```

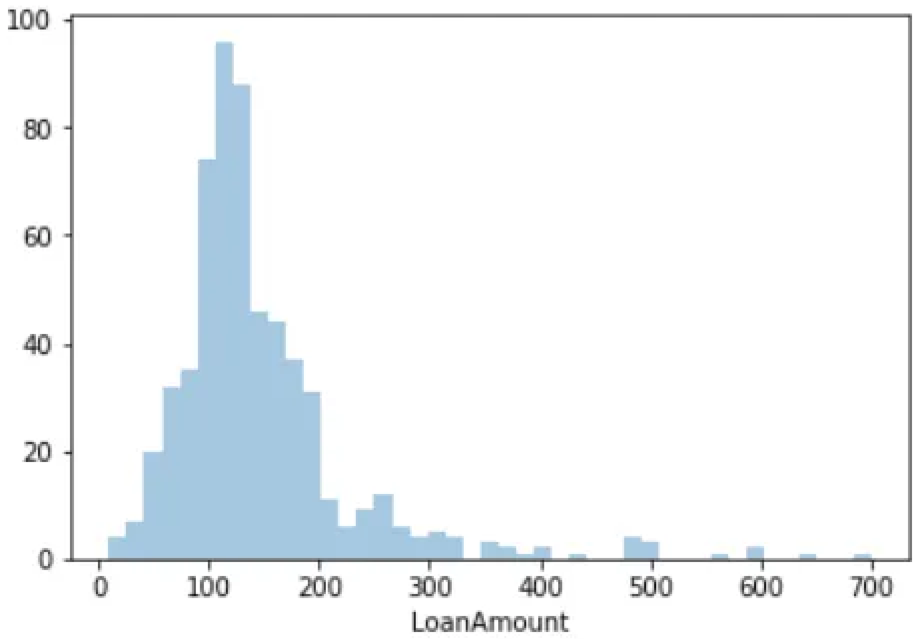


Image by Author

People with better education should normally have a higher income, we can check that by plotting the education level against the income.

```
sns.boxplot(x='Education',y='ApplicantIncome',data=train)
```

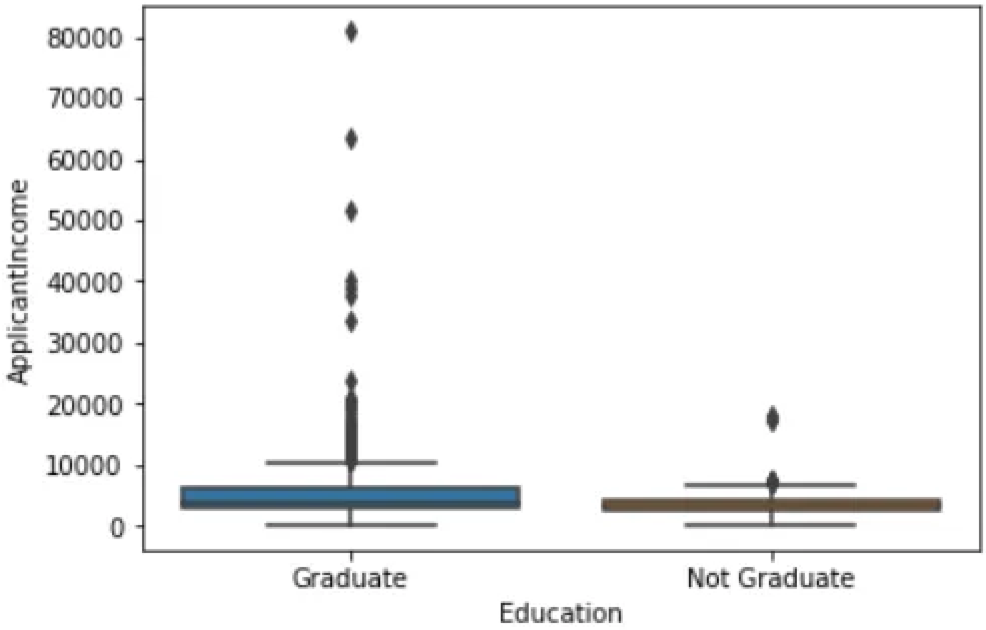


Image by Author

The distributions are quite similar but we can see that the graduates have more outliers which means that the people with huge income are most likely well educated.

Another interesting variable is credit history , to check how it affects the Loan Status we can turn it into binary then calculate it's mean for each value of credit history . A value close to 1 indicates a high loan success rate

```
#turn loan status into binary
modified=train
modified['Loan_Status']=train['Loan_Status'].apply(lambda x: 0 if
x=="N" else 1 )
#calculate the mean
modified.groupby('Credit_History').mean()['Loan_Status']

OUT :
Credit_History
0.0    0.078652
1.0    0.795789
Name: Loan_Status, dtype: float64
```

People with a credit history a way more likely to pay their loan, 0.07 vs 0.79 . This means that credit history will be an influential variable in our model.

Data preprocessing:

The first thing to do is to deal with the missing value , lets check first how many there are for each variable.

```
train.apply(lambda x: sum(x.isnull()),axis=0)
OUT:
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

For numerical values a good solution is to fill missing values with the mean , for categorical we can fill them with the mode (the value with the highest frequency)

```
#categorical
train['Gender'].fillna(train['Gender'].mode()[0], inplace=True)
train['Married'].fillna(train['Married'].mode()[0], inplace=True)
train['Dependents'].fillna(train['Dependents'].mode()[0],
inplace=True)
train['Loan_Amount_Term'].fillna(train['Loan_Amount_Term'].mode()[0],
inplace=True)
train['Credit_History'].fillna(train['Credit_History'].mode()[0],
inplace=True)
train['Self_Employed'].fillna(train['Self_Employed'].mode()[0],
inplace=True)
#numerical

df['LoanAmount'].fillna(df['LoanAmount'].mean(), inplace=True)
```

Next we have to handle the outliers , one solution is just to remove them but we can also log transform them to nullify their effect which is the approach that we went for here. Some people might have a low income but strong

CoappliantIncome so a good idea is to combine them in a TotalIncome column.

```
train['LoanAmount_log']=np.log(train['LoanAmount'])
train['TotalIncome']= train['ApplicantIncome']
+train['CoapplicantIncome']
train['TotalIncome_log']=np.log(train['TotalIncome'])
```

plotting the histogram of loan amount log we can see that it's a normal distribution!

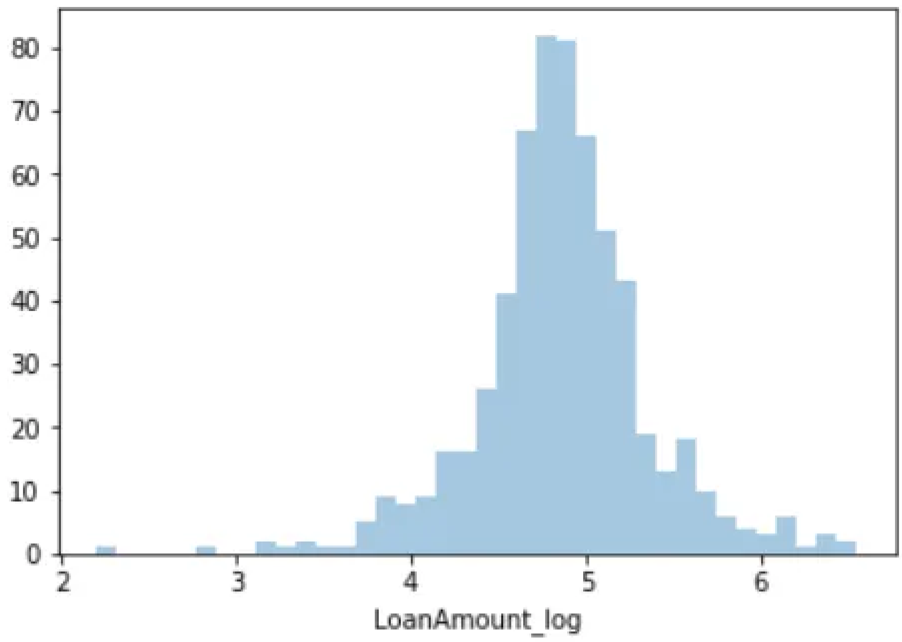


Image by Author

Modeling:

We're gonna use sklearn for our models , before doing that we need to turn all the categorical variables into numbers. We'll do that using the LabelEncoder in sklearn

```
from sklearn.preprocessing import LabelEncoder
category=
['Gender','Married','Dependents','Education','Self_Employed','Propert
y_Area','Loan_Status']
encoder= LabelEncoder()
 for i in category:
```

```
    train[i] = encoder.fit_transform(train[i])
    train.dtypes

OUT:
Loan_ID                  object
Gender                    int64
Married                   int64
Dependents                int64
Education                 int64
Self_Employed             int64
ApplicantIncome           int64
CoapplicantIncome       float64
LoanAmount              float64
Loan_Amount_Term        float64
Credit_History          float64
Property_Area             int64
Loan_Status               int64
LoanAmount_log          float64
TotalIncome             float64
TotalIncome_log         float64
dtype: object
```

Now all our variables have became numbers that our models can understand.

To try out different models we'll create a function that takes in a model , fits it and mesures the accuracy which means using the model on the train set and mesuring the error on the same set . And we'll use a technique called Kfold cross validation which splits randomly the data into train and test set, trains the model using the train set and validates it with the test set, it will repeat this K times hence the name Kfold and takes the average error. The latter method gives a better idea on how the model performs in real life.

```
#Import the models
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import KFold   #For K-fold cross
validation
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn import metrics

def classification_model(model, data, predictors, outcome):
  #Fit the model:
  model.fit(data[predictors],data[outcome])

  #Make predictions on training set:
  predictions = model.predict(data[predictors])
```

```
    #Print accuracy
    accuracy = metrics.accuracy_score(predictions,data[outcome])
    print ("Accuracy : %s" % "{0:.3%}".format(accuracy))

  #Perform k-fold cross-validation with 5 folds
    kf = KFold(data.shape[0], n_folds=5)
    error = []
    for train, test in kf:
      # Filter training data
      train_predictors = (data[predictors].iloc[train,:])

      # The target we're using to train the algorithm.
      train_target = data[outcome].iloc[train]

      # Training the algorithm using the predictors and target.
      model.fit(train_predictors, train_target)

      #Record error from each cross-validation run
      error.append(model.score(data[predictors].iloc[test,:],
  data[outcome].iloc[test]))

    print ("Cross-Validation Score : %s" % "
  {0:.3%}".format(np.mean(error)))
```

Now we can test different models we'll start with logistic regression:

```
outcome_var = 'Loan_Status'
model = LogisticRegression()
predictor_var =
['Credit_History','Education','Married','Self_Employed','Property_Are
a']
classification_model(model, train,predictor_var,outcome_var)
OUT :
Accuracy : 80.945%
Cross-Validation Score : 80.946%
```

We'll try now a Decision tree which is should give us more accurate result

```
model = DecisionTreeClassifier() predictor_var =
['Credit_History','Gender','Married','Education']
classification_model(model, df,predictor_var,outcome_var)

OUT:
Accuracy : 80.945%
Cross-Validation Score : 78.179%
```

We’ve got the same score on accuracy but a worse score in cross validation , a more complex model doesn’t always means a better score.

Finally we’ll try random forests

```
model = RandomForestClassifier(n_estimators=100)
predictor_var = ['Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'Loan_Amount_Term', 'Credit_History',
'Property_Area',
        'LoanAmount_log','TotalIncome_log']
classification_model(model, train,predictor_var,outcome_var)

OUT:
Accuracy : 100.000%
Cross-Validation Score : 78.015%
```

The model is giving us perfect score on accuracy but a low score in cross validation , this a good example of over fitting. The model is having a hard time at generalizing since it’s fitting perfectly to the train set.

Solutions to this include : Reducing the number of predictors or Tuning the model parameters.

Conclusion:

We’ve gone through a good portion of the data science pipe line in this article, namely EDA , preprocessing and modeling and we’ve used essential classification models such as Logistic regression , Decision tree and Random forests. It would be interesting to learn more about the backbone logic behind these algorithms, and also tackle the data scraping and deployment phases.We’ll try to do that in the next articles.

Machine Learning Data Science Data Visualization Computer Science

Written by Tariq Massaoudi

Follow

132 Followers · Writer for Towards Data Science

AI and machine learning enthusiast. https://www.linkedin.com/in/tariqmassaoudi

---

More from Tariq Massaoudi and Towards Data Science

Tariq Massaoudi

How I Created an AI Clone of Myself With ChatGPT

Using Lang Chain & Pinecone to create a chatbot that simulates my personality

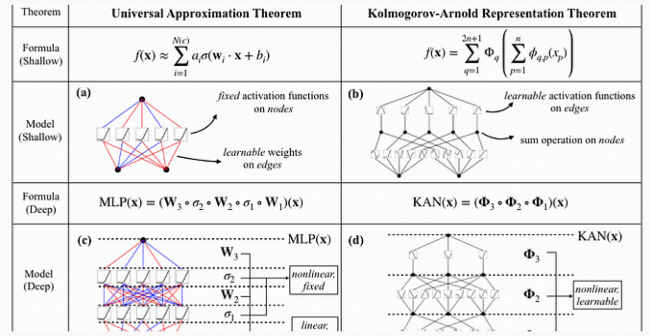
Apr 21, 2023 · 94 · 2

Torsten Walbaum in Towards Data Science

What 10 Years at Uber, Meta and Startups Taught Me About Data...

Advice for Data Scientists and Managers

May 31 · 5.1K · 82



Theo Wolf in Towards Data Science

Kolmogorov-Arnold Networks: the latest advance in Neural Network...

Tariq Massaoudi

How I Passed The AWS Solution Architect Associate (SAA-C03)

The new type of network that is making waves in the ML world.

May 13 · 2.1K · 19

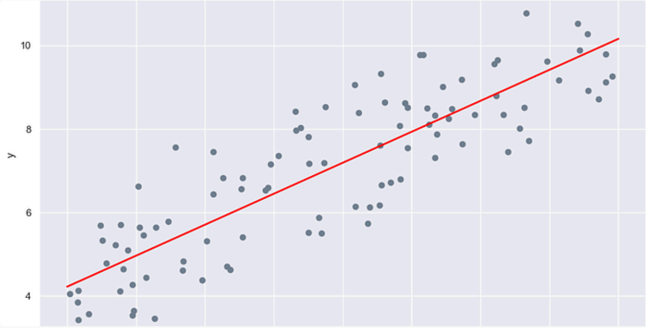
I passed the AWS Solutions Architect Associate (SSA-003) exam in December…

Dec 2, 2022 · 153 · 3

See all from Tariq Massaoudi

See all from Towards Data Science

Recommended from Medium

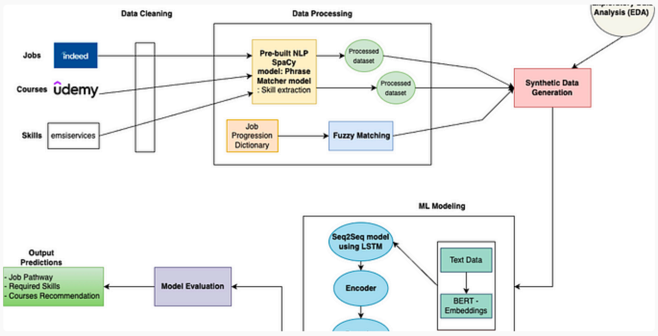


Rakesh Ganya

From Data to Predictions: Understanding Linear Regression

Linear Regression

Jun 6 · 10



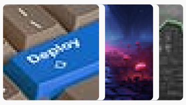
Nitin Mali

Predicting Job Titles, Skills, and Recommending Online Courses…

Introduction:

Jan 4 · 51

Lists

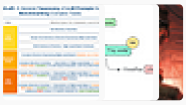


Predictive Modeling w/ Python

20 stories · 1297 saves

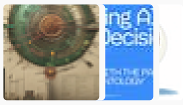
Practical Guides to Machine Learning

10 stories · 1561 saves



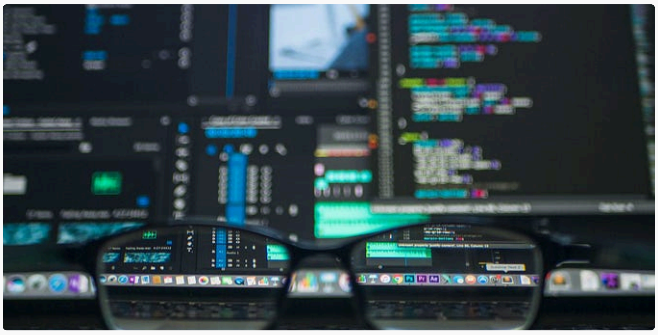
Natural Language Processing

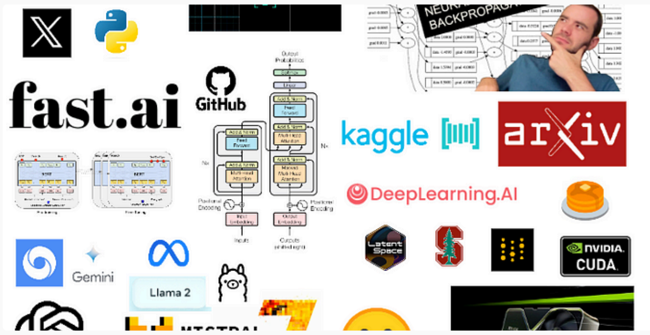
1518 stories · 1053 saves



data science and AI

40 stories · 185 saves





Bart Zalewski

Integrating Machine Learning into Web Development

Introduction

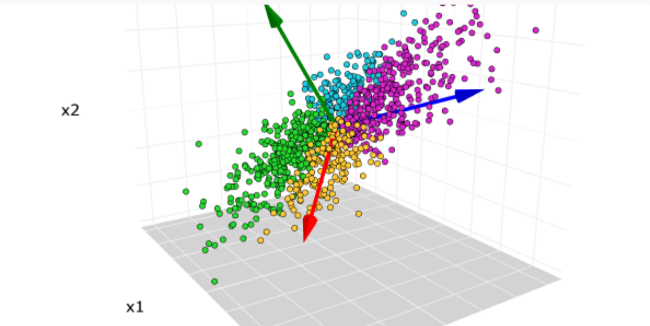
Jan 9 10

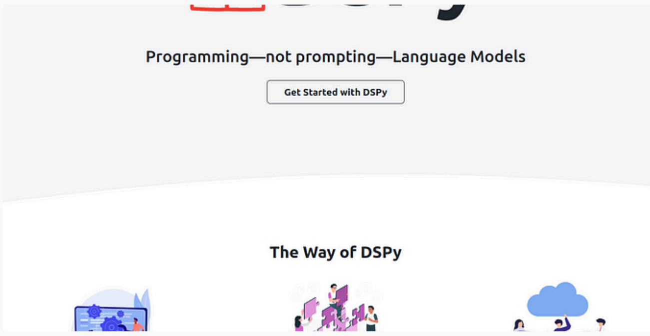
Benedict Neo in bitgrit Data Science Publication

Roadmap to Learn AI in 2024

A free curriculum for hackers and programmers to learn AI

Mar 11 12.5K 136





Kavishka Abeywardana

Principal Component Analysis (PCA)

2d ago 13

Vishal Rajput in AIGuys

Prompt Engineering Is Dead: DSPy Is New Paradigm For Prompting

DSPy Paradigm: Let's program—not prompt—LLMs

May 29 3K 32

See more recommendations