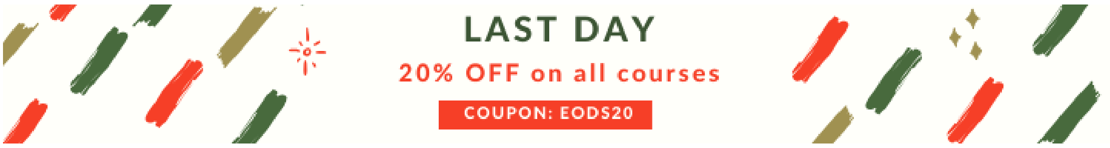


BUSINESS ANALYTICS INTERMEDIATE MACHINE LEARNING PYTHON RESEARCH & TECHNOLOGY STRUCTURED DATA TECHNIQUE

Introductory guide to Linear Optimization in Python (with TED videos case study)

GUEST BLOG, OCTOBER 9, 2017 LOGIN TO BOOKMARK THIS ARTICLE

Introduction

Data Science & Machine Learning are being used by organizations to solve a variety of business problems today. In order to create a real business impact, an important consideration is to bridge the gap between the data science pipeline and business decision making pipeline.

The outcome of data science pipeline is uaully predictions, patterns and insights from data (typically without any notion of constraints) but that alone is insufficient for business stakeholders to take decisions. Data science output has to be fed into the business decision making pipeline which involves some sort of optimization involving constraints and decision variables which model key aspects of the business.

For example, if you are running a Super Market chain – your data science pipeline would forecast the expected sales. You would then take those inputs and create an optimised inventory / sales strategy.

In this article, we will show one such example of Linear optimization for selecting which TED videos to watch.

Table of Contents

Introduction to Linear Optimization

Among optimization techniques, Linear Optimization using the Simplex Method is considered one of the most powerful ones and has been rated as one of the Top 10 algorithms of the 20th century. As data science practitioners, it is important to have hands-on knowledge in implementing Linear Optimization and this blog post is to illustrate its implementation using Python's PuLP package.

To make things interesting & simpler to understand, we will learn this optimization technique by applying it on a practical, day-to-day problem. Having said that, what we learn is applicable to a variety of business problems as well.

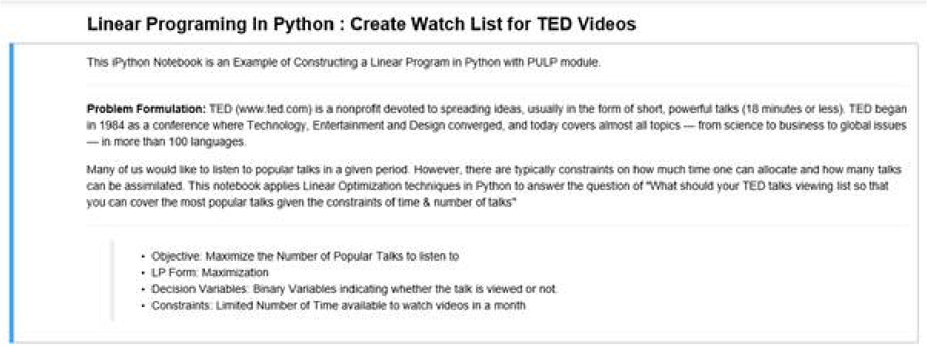
Note: This article assumes you have a basics knowledge of linear programming. You can go through this article if you want to review the topic.

The Problem – Creating the Watch List for TED videos

TED is a nonprofit devoted to spreading ideas. TED began in 1984 as a conference where Technology, Entertainment and Design converged, and today covers almost all topics — from science to business to global issues — in more than 100 languages. TED talks are delivered by experts passionate about work in their chosen domains and have a wealth of information.

Now, for the purpose of this blog post, imagine a situation where one is interested to create their watch list of the most popular TED talks given their constraints (time that can be allotted to viewing and the number of talks). We will see how to implement the Python program to help us create the watchlist in the optimal manner.

The code of the article can be found here. Screenshots from my Jupyter notebook are shown below:



Linear Programing In Python : Create Watch List for TED Videos

This iPython Notebook is an Example of Constructing a Linear Program in Python with PULP module.

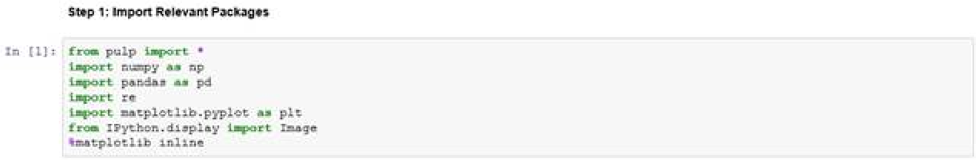
Problem Formulation: TED (www.ted.com) is a nonprofit devoted to spreading ideas, usually in the form of short, powerful talks (18 minutes or less). TED began in 1984 as a conference where Technology, Entertainment and Design converged, and today covers almost all topics — from science to business to global issues — in more than 100 languages.

Many of us would like to listen to popular talks in a given period. However, there are typically constraints on how much time one can allocate and how many talks can be assimilated. This notebook applies Linear Optimization techniques in Python to answer the question of "What should your TED talks viewing list so that you can cover the most popular talks given the constraints of time & number of talks"

> - Objective: Maximize the Number of Popular Talks to listen to
> - LP Form: Maximization
> - Decision Variables: Binary Variables indicating whether the talk is viewed or not.
> - Constraints: Limited Number of Time available to watch videos in a month

Step 1 – Import relevant packages

PuLP is a free open source software written in Python. It is used to describe optimisation problems as mathematical models. PuLP can then call any of numerous external LP solvers (CBC, GLPK, CPLEX, Gurobi etc) to solve this model and then use python commands to manipulate and display the solution. By default, CoinMP solver is bundled with PuLP.

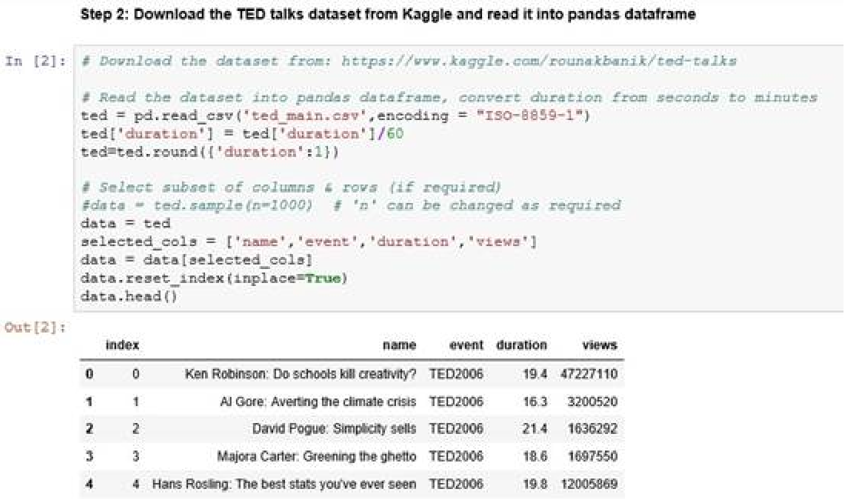


Step 1: Import Relevant Packages

```
In [1]: from pulp import *
        import numpy as np
        import pandas as pd
        import re
        import matplotlib.pyplot as plt
        from IPython.display import Image
        %matplotlib inline
```

Step 2 – Create a dataframe for TED talks

Dataset having all the TED talks (2550) is downloaded from Kaggle and read into a dataframe. A subset of relevant columns is selected and the resulting dataset has the following details – Index of the talk, Name of the talk, TED Event Name, Talk duration (in minutes), Number of Views (Proxy for Popularity of the talk)



Step 2: Download the TED talks dataset from Kaggle and read it into pandas dataframe

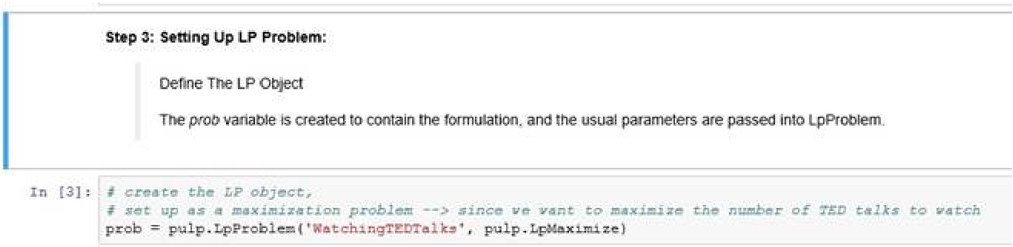
```
In [2]: # Download the dataset from: https://www.kaggle.com/rounakbanik/ted-talks

        # Read the dataset into pandas dataframe, convert duration from seconds to minutes
        ted = pd.read_csv('ted_main.csv',encoding = "ISO-8859-1")
        ted['duration'] = ted['duration']/60
        ted=ted.round({'duration':1})

        # Select subset of columns & rows (if required)
        #data = ted.sample(n=1000)   # 'n' can be changed as required
        data = ted
        selected_cols = ['name','event','duration','views']
        data = data[selected_cols]
        data.reset_index(inplace=True)
        data.head()
```

Out[2]:

	index	name	event	duration	views
0	0	Ken Robinson: Do schools kill creativity?	TED2006	19.4	47227110
1	1	Al Gore: Averting the climate crisis	TED2006	16.3	3200520
2	2	David Pogue: Simplicity sells	TED2006	21.4	1636292
3	3	Majora Carter: Greening the ghetto	TED2006	18.6	1697550
4	4	Hans Rosling: The best stats you've ever seen	TED2006	19.8	12005869

Step 3 – Set up the Linear Optimization Problem

Start with defining the LP Object. The prob variable is created to contain the problem formulation



Step 3: Setting Up LP Problem:

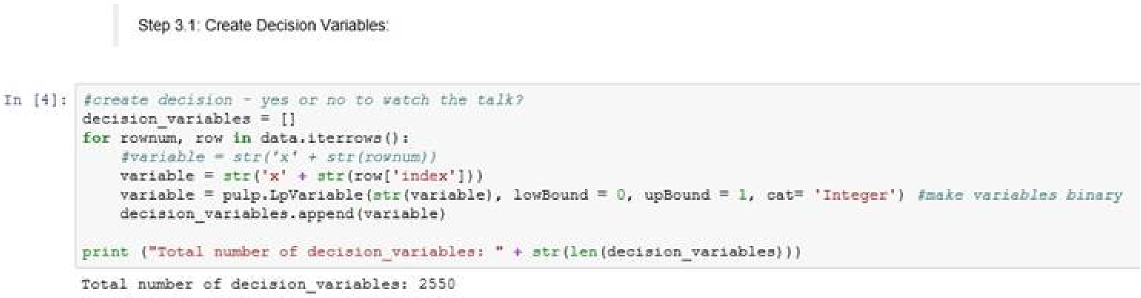
Define The LP Object

The *prob* variable is created to contain the formulation, and the usual parameters are passed into LpProblem.

```
In [3]: # create the LP object,
        # set up as a maximization problem --> since we want to maximize the number of TED talks to watch
        prob = pulp.LpProblem('WatchingTEDTalks', pulp.LpMaximize)
```

Step 3.1: Create the decision variables

Iterate over each row of the data frame to create the decision variables, such that each talk becomes one decision variable. Since each talk can either be selected or not selected as part of the final watch list, the decision variable is binary in nature (1=Selected, 0=Not Selected)

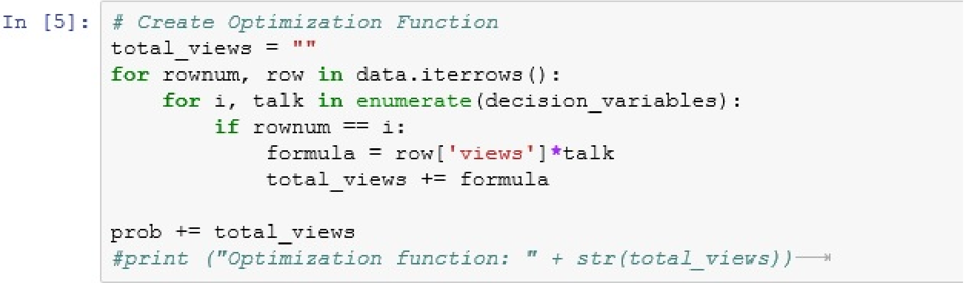


Step 3.1: Create Decision Variables:

```
In [4]: #create decision - yes or no to watch the talk?
        decision_variables = []
        for rownum, row in data.iterrows():
            #variable = str('x' + str(rownum))
            variable = str('x' + str(row['index']))
            variable = pulp.LpVariable(str(variable), lowBound = 0, upBound = 1, cat= 'Integer') #make variables binary
            decision_variables.append(variable)

        print ("Total number of decision_variables: " + str(len(decision_variables)))

        Total number of decision_variables: 2550
```

Step 3.2: Define the Objective Function

The objective function is the sum over all rows of the views for each talk. The views serve as a proxy for the popularity of the talk and so in essence we are trying to maximize the views (popularity) by selecting appropriate talks (decision variables)



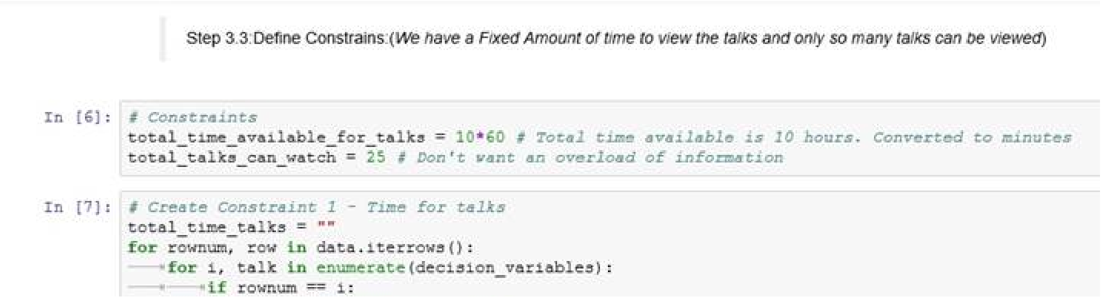
```
In [5]: # Create Optimization Function
        total_views = ""
        for rownum, row in data.iterrows():
            for i, talk in enumerate(decision_variables):
                if rownum == i:
                    formula = row['views']*talk
                    total_views += formula

        prob += total_views
        #print ("Optimization function: " + str(total_views))
```

Step 3.3: Define the Constraints

In the problem, we have 2 constraints:

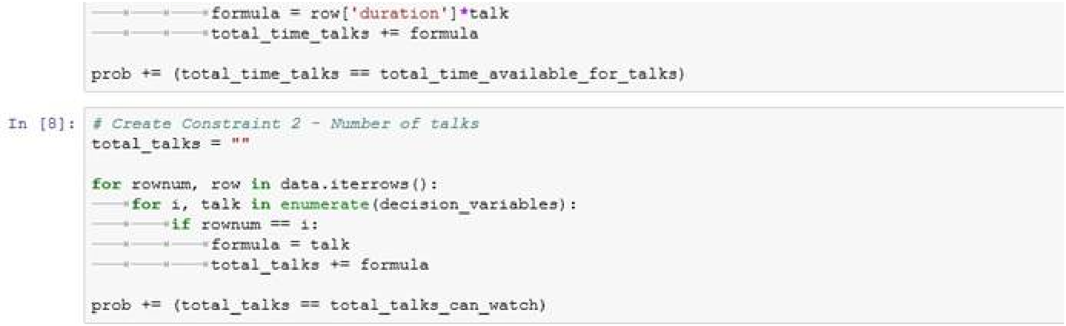
a) We only have fixed amount of total time that can be allocated to view the talks

b) We don’t want to view more than a certain number of talks to avoid information overload



Step 3.3:Define Constrains:(*We have a Fixed Amount of time to view the talks and only so many talks can be viewed*)

```
In [6]: # Constraints
        total_time_available_for_talks = 10*60 # Total time available is 10 hours. Converted to minutes
        total_talks_can_watch = 25 # Don't want an overload of information

In [7]: # Create Constraint 1 - Time for talks
        total_time_talks = ""
        for rownum, row in data.iterrows():
            for i, talk in enumerate(decision_variables):
                if rownum == i:
```

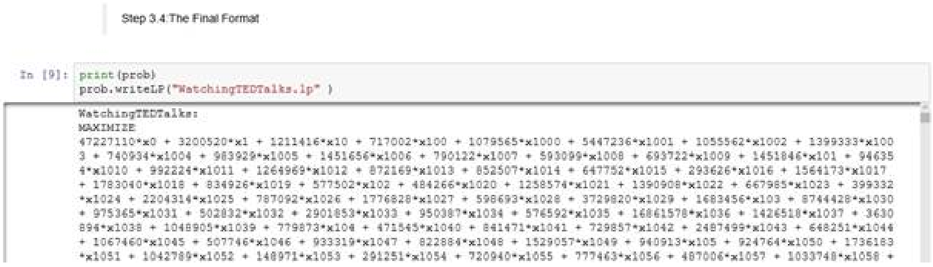


```
            formula = row['duration']*talk
            total_time_talks += formula

prob += (total_time_talks == total_time_available_for_talks)
```

```
In [8]: # Create Constraint 2 - Number of talks
        total_talks = ""

        for rownum, row in data.iterrows():
            for i, talk in enumerate(decision_variables):
                if rownum == i:
                    formula = talk
                    total_talks += formula

        prob += (total_talks == total_talks_can_watch)
```

Step 3.4: The Final Format (for problem formulation)

The final format of the problem formulated is written out into a .lp file. This will list the objective function, the decision variables and the constraints imposed on the problem.



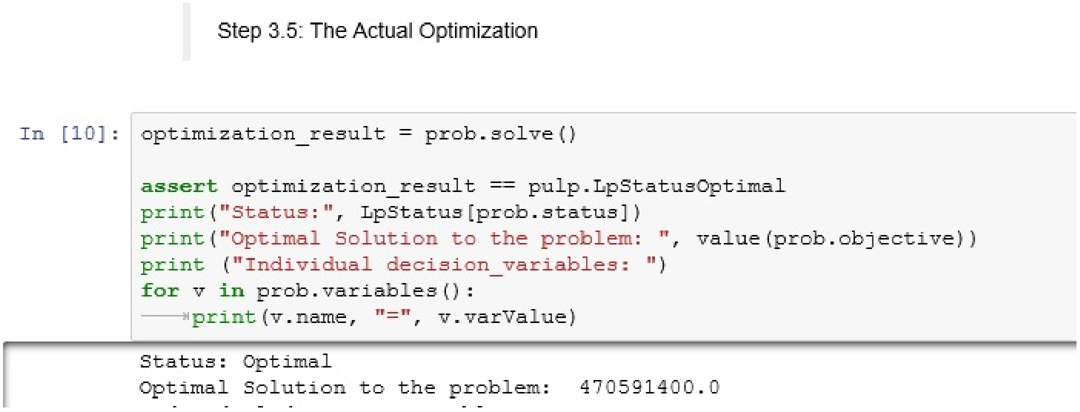
Step 3.4 The Final Format

```
In [9]: print(prob)
        prob.writeLP("WatchingTEDTalks.lp" )
```

```
WatchingTEDTalks:
MAXIMIZE
47227110*x0 + 3200520*x1 + 1211416*x10 + 717002*x100 + 1079565*x1000 + 5447236*x1001 + 1055562*x1002 + 1399333*x100
3 + 740934*x1004 + 983929*x1005 + 1451656*x1006 + 790122*x1007 + 593099*x1008 + 693722*x1009 + 1451846*x101 + 94635
4*x1010 + 992224*x1011 + 1264969*x1012 + 872169*x1013 + 852507*x1014 + 647752*x1015 + 293626*x1016 + 1564173*x1017
+ 1783040*x1018 + 834926*x1019 + 577502*x102 + 484266*x1020 + 1258574*x1021 + 1390908*x1022 + 667985*x1023 + 399332
*x1024 + 2204314*x1025 + 787092*x1026 + 1776828*x1027 + 598693*x1028 + 3729820*x1029 + 1683456*x103 + 8744428*x1030
+ 975365*x1031 + 502832*x1032 + 2901853*x1033 + 950387*x1034 + 576592*x1035 + 16861578*x1036 + 1426518*x1037 + 3630
894*x1038 + 1048905*x1039 + 779873*x104 + 471545*x1040 + 841471*x1041 + 729857*x1042 + 2487499*x1043 + 648251*x1044
+ 1067460*x1045 + 507746*x1046 + 933319*x1047 + 822884*x1048 + 1529057*x1049 + 940913*x105 + 924764*x1050 + 1736183
*x1051 + 1042789*x1052 + 148971*x1053 + 291251*x1054 + 720940*x1055 + 777463*x1056 + 487006*x1057 + 1033748*x1058 +
```

Step 3.5: The Actual Optimization

The actual optimization is a single line of code that calls 'prob.solve'. Assert statement is inserted to ascertain whether an optimal result was obtained for the problem.



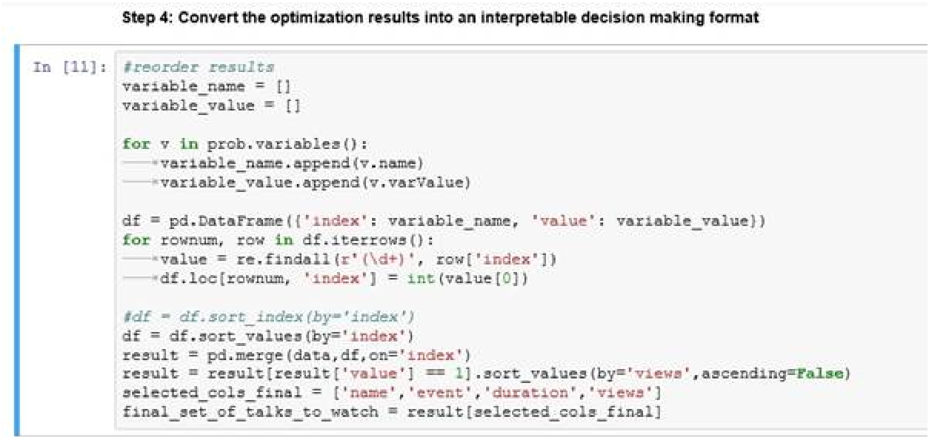
Step 3.5: The Actual Optimization

```
In [10]: optimization_result = prob.solve()

         assert optimization_result == pulp.LpStatusOptimal
         print("Status:", LpStatus[prob.status])
         print("Optimal Solution to the problem: ", value(prob.objective))
         print ("Individual decision_variables: ")
         for v in prob.variables():
             print(v.name, "=", v.varValue)
```

```
Status: Optimal
Optimal Solution to the problem:  470591400.0
```

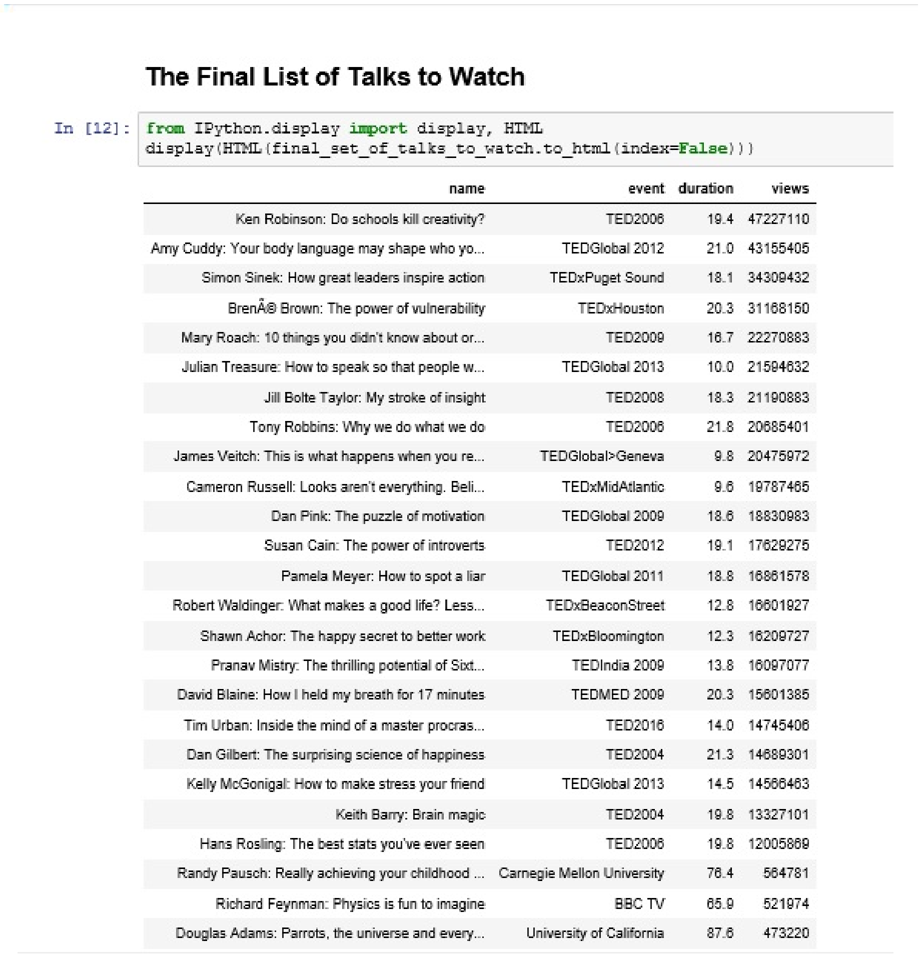
Step 4 – Convert the Optimization results into an interpretable format

The optimization results which indicates the specific decision variables (talks) that were selected to maximize the outcome has to be converted into a format of a watch list, as shown below:



Step 4: Convert the optimization results into an interpretable decision making format

```
In [11]: #reorder results
         variable_name = []
         variable_value = []

         for v in prob.variables():
             variable_name.append(v.name)
             variable_value.append(v.varValue)

         df = pd.DataFrame({'index': variable_name, 'value': variable_value})
         for rownum, row in df.iterrows():
             value = re.findall(r'(\d+)', row['index'])
             df.loc[rownum, 'index'] = int(value[0])

         #df = df.sort_index(by='index')
         df = df.sort_values(by='index')
         result = pd.merge(data,df,on='index')
         result = result[result['value'] == 1].sort_values(by='views',ascending=False)
         selected_cols_final = ['name','event','duration','views']
         final_set_of_talks_to_watch = result[selected_cols_final]
```



The Final List of Talks to Watch

```
In [12]: from IPython.display import display, HTML
         display(HTML(final_set_of_talks_to_watch.to_html(index=False)))
```

name	event	duration	views
Ken Robinson: Do schools kill creativity?	TED2006	19.4	47227110
Amy Cuddy: Your body language may shape who yo...	TEDGlobal 2012	21.0	43155405
Simon Sinek: How great leaders inspire action	TEDxPuget Sound	18.1	34309432
BrenÃ© Brown: The power of vulnerability	TEDxHouston	20.3	31168150
Mary Roach: 10 things you didn't know about or...	TED2009	16.7	22270883
Julian Treasure: How to speak so that people w...	TEDGlobal 2013	10.0	21594632
Jill Bolte Taylor: My stroke of insight	TED2008	18.3	21190883
Tony Robbins: Why we do what we do	TED2006	21.8	20685401
James Veitch: This is what happens when you re...	TEDGlobal>Geneva	9.8	20475972
Cameron Russell: Looks aren't everything. Beli...	TEDxMidAtlantic	9.6	19787465
Dan Pink: The puzzle of motivation	TEDGlobal 2009	18.6	18830983
Susan Cain: The power of introverts	TED2012	19.1	17629275
Pamela Meyer: How to spot a liar	TEDGlobal 2011	18.8	16861578
Robert Waldinger: What makes a good life? Less...	TEDxBeaconStreet	12.8	16601927
Shawn Achor: The happy secret to better work	TEDxBloomington	12.3	16209727
Pranav Mistry: The thrilling potential of Sixt...	TEDIndia 2009	13.8	16097077
David Blaine: How I held my breath for 17 minutes	TEDMED 2009	20.3	15601385
Tim Urban: Inside the mind of a master procras...	TED2016	14.0	14745406
Dan Gilbert: The surprising science of happiness	TED2004	21.3	14689301
Kelly McGonigal: How to make stress your friend	TEDGlobal 2013	14.5	14566463
Keith Barry: Brain magic	TED2004	19.8	13327101
Hans Rosling: The best stats you've ever seen	TED2006	19.8	12005869
Randy Pausch: Really achieving your childhood ...	Carnegie Mellon University	76.4	564781
Richard Feynman: Physics is fun to imagine	BBC TV	65.9	521974
Douglas Adams: Parrots, the universe and every...	University of California	87.6	473220

End Notes

This article provides an example of utilizing Linear Optimization techniques available in Python to solve the everyday problem of creating video watch list. The concepts learned are also applicable in more complex business situations involving thousands of decision variables and many different constraints.

Every data science practitioner needs to add “Optimization techniques” to their body of knowledge so that they can use advanced analytics to solve real world business problems and this article is intended to help you take the first step in that direction.

Karthikeyan Sankaran is currently a Director at LatentView Analytics which provides solutions at the intersection of Business, Technology & Math to business problems across a wide range of industries. Karthik has close to two decades of experience in the Information Technology industry having worked in multiple roles across the space of Data Management, Business Intelligence & Analytics.

This story was received as part of “Blogathon” contest on Analytics Vidhya. Karthikeyan’s entry was one of the winning entries in the competition.

Share this:

Like this:

Loading...

TAGS : LINEAR OPTIMIZATION, PANDAS, PULP, WATCHLIST FOR TED TALKS

NEXT ARTICLE

8 Essential Tips for People starting a Career in Data Science

•••

PREVIOUS ARTICLE

25 Questions to test a Data Scientist on Image Processing

Guest Blog

This article is quite old and you might not get a prompt response from the author. We request you to post this comment on Analytics Vidhya's Discussion portal to get your queries resolved

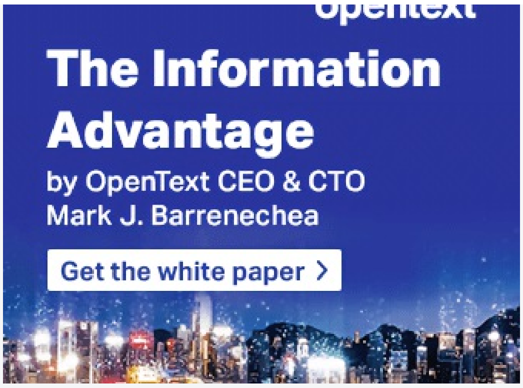
ONE COMMENT

PRAMODTATA184

October 10, 2017 at 2:37 pm

Reply

Very useful. Thank you sir.



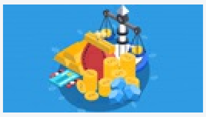
RECOMMENDED READS

A Complete Python Tutorial to Learn Data Science from Scratch

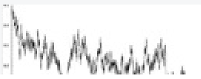
Commonly used Machine Learning Algorithms (with Python and R Codes)

7 Regression Techniques you should know!

RECOMMENDED RESOURCES

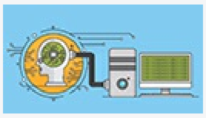

Practice & Learn

Loan Prediction



Free Course for you

Creating Time Series Forecast using Python



Try for free

Applied Machine Learning - Beginner to Professional

Analytics Vidhya

Learn everything about analytics

Download App

Analytics Vidhya

About Us

Our Team

Careers

Contact us

Data Science

Blog

Hackathon

Discussions

Apply Jobs

Companies

Post Jobs

Trainings

Hiring Hackathons

Advertising

Visit us

Privacy Policy| Terms of Use| Refund Policy