ทำนายคะแนนต่ำสุด วิศวะ จุฬา TCAS รอบ 3 (รุ่น Dek63)

โดย พี่เกรท ออนดีมานด์

โจทย์: ทำนายคะแนนต่ำสุด สอบติดวิศวะ จุฬา TCAS รอบ 3 ด้วย Machine Learning Models

> **Disclaimer:** การวิเคราะห์นี้ ทำด้วยข้อมูลเท่าที่หาได้จากเว็บไซต์ สทศ ซึ่งมีข้อมูลย้อนหลังถึงปี 2553 ที่เริ่มมีการสอบ GAT/PAT อีกทั้งเราไม่เอาคะแนนปี 53-54 มาคิดเพราะเกณฑ์การรับมี GPAX ด้วย ต่างจากปี 55 เป็นต้นไป ที่ใช้คะแนน GAT/PAT เพียงอย่างเดียว ดังนั้น จำนวนจุดของข้อมูลจึงมีจำกัดมากๆ อาจทำให้คะแนนที่ทำนายออกมา มีความแม่นยำที่ลดลง อย่างไรก็ตาม พี่จะทำนายแล้วระบุความคลาดเคลื่อนเอาไว้ให้ด้วยครับ

TL;DR

จากการทำนายด้วยโมเดล Linear Regression พบว่า คะแนนทำนายต่ำสุดเพื่อสอบติด วิศวะ จุฬา TCAS รอบ 3 ปี 2563 โดยใช้ข้อมูลปี 2555-2562 คือ

> **21,112 ± 1,072 คะแนน**

ข้อสรุปจากการวิเคราะห์

- คะแนนปีนี้ มีแนวโน้ม **เฟ้อ**มากกว่าทุกปีที่ผ่านมา (สังเกตเบื้องต้นได้จากจำนวนคนทำ PAT1 ได้เกิน 120 ที่เพิ่มขึ้นจากปีก่อนๆ ถึงเท่าตัว)
- คะแนนปีนี้ มีแนวโน้มใกล้เคียงคะแนนปี **2560** และ **2561**
- น้องที่มีคะแนนในช่วง 20,000 มีสิทธิ์ลุ้นสอบติดนะครับ (แม้แต่หากน้องได้ 19,XXX ก็ควรลองยื่นเป็นอันดับ 1 ดูครับ)
- แต่พี่ขอย้ำอีกทีนะครับว่า คะแนนทำนายนี้ เป็นเพียงคะแนนทำนายเท่านั้น ทำให้อาจมีปัจจัยอื่นที่โมเดลไม่ได้คำนึงถึงครับ

ทั้งนี้ การเลือกคณะ พี่แนะนำให้น้องเลือกด้วยหลักการดังนี้นะครับ

- **อันดับ 1:** เลือกตามความอยากเลยโดยไม่ต้องแคร์คะแนนที่น้องได้ จะได้ไม่คาใจครับ
- **อันดับ 2-3:** คะแนนของน้อง อยู่ในช่วงค่าต่ำสุดที่สอบติดของ 3 ปีย้อนหลัง
- **อันดับ 4-5:** คะแนนของน้อง เกินคะแนนต่ำสุดที่สอบติดของ 3 ปี ย้อนหลัง อย่างน้อย 1500 คะแนน (สเกลเต็ม 30000)
- **อันดับ 6:** คะแนนของน้อง เกินคะแนนต่ำสุดที่สอบติดของ 3 ปี ย้อนหลัง อย่างน้อย 3000 คะแนน (สเกลเต็ม 30000)

และน้องสามารถวิเคราะห์คะแนนและดูสถิติเพิ่มเติม ได้ที่แอพพลิเคชั่น TCASter (https://tcaster.net/ (https://tcaster.net/)) หรือติดตามข่าวสารได้ที่ Facebook Fanpage (https://www.facebook.com/TCASterApp/ (https://www.facebook.com/TCASterApp/)) ครับ

แล้วก็ เดี๋ยวพี่จะวิเคราะห์คะแนน กสพท ให้ด้วย รอติดตามได้ทาง Instagram ของพี่ ที่ https://www.instagram.com/pgreatondemand (https://www.instagram.com/pgreatondemand) นะครับ

ขอให้น้องๆ สอบติดได้ตามที่ตั้งใจไว้นะ :-)

Data Import

In [1]:
```
import pandas as pd
import numpy as np

df = pd.read_csv('engcuraw63_04.csv', index_col=0)
```

In [2]:
```
df.head()
```

Out[2]:

Year	Round	CU_R3_Min	Quota	GAT_N	PAT1_N	PAT3_N	GAT_avg	GAT_std	PAT1_avg	PAT1_std	..
2555	1	16465.0	800	284735	200693	39488	130.59	68.04	39.64	20.07	..
2556	1	19100.0	800	323876	243834	51235	114.30	60.63	40.61	20.96	..
2557	1	18552.0	600	237419	173708	35302	143.58	61.60	57.40	24.13	..
2558	1	19090.0	610	316791	239345	49776	131.97	63.81	51.56	18.85	..
2559	1	17510.0	610	324856	253213	57879	115.66	58.73	52.61	21.64	..

5 rows × 35 columns

In [3]:
```
X = df.drop('CU_R3_Min', axis=1)
y = df[['CU_R3_Min']]
```

In [4]:
```
X
```

Out[4]:

Year	Round	Quota	GAT_N	PAT1_N	PAT3_N	GAT_avg	GAT_std	PAT1_avg	PAT1_std	PAT3_avg	...
2555	1	800	284735	200693	39488	130.59	68.04	39.64	20.07	83.45	...
2556	1	800	323876	243834	51235	114.30	60.63	40.61	20.96	91.11	...
2557	1	600	237419	173708	35302	143.58	61.60	57.40	24.13	99.40	...
2558	1	610	316791	239345	49776	131.97	63.81	51.56	18.85	93.41	...
2559	1	610	324856	253213	57879	115.66	58.73	52.61	21.64	80.66	...
2560	1	590	289866	228522	50623	123.93	58.48	42.82	25.34	95.91	...
2561	1	610	230566	177835	37347	144.65	67.82	48.45	26.13	92.32	...
2562	1	430	215585	163251	34493	144.06	66.90	49.05	29.68	90.81	...
2563	1	400	202341	150694	33147	147.50	63.43	62.90	38.08	98.21	...

9 rows × 34 columns

In [5]:
```
y
```

Out[5]:

Year	CU_R3_Min
2555	16465.0
2556	19100.0
2557	18552.0
2558	19090.0
2559	17510.0
2560	20773.0
2561	20570.0
2562	18360.0
2563	NaN

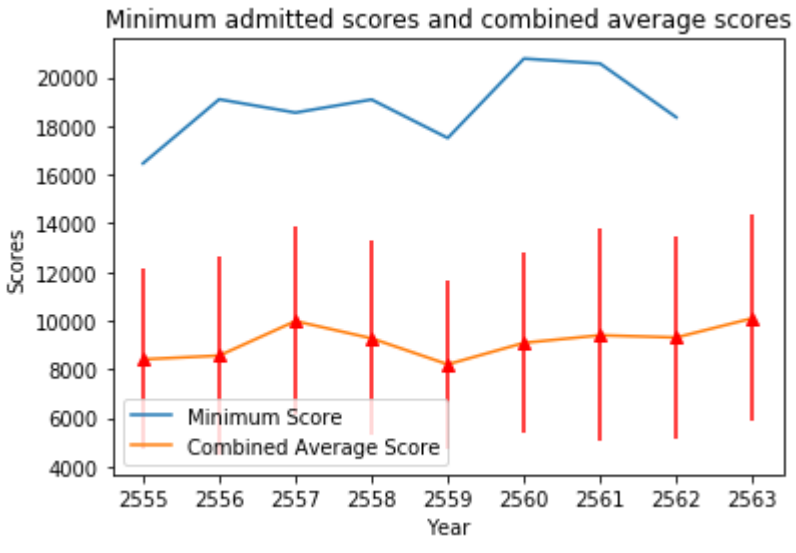
Exploratory Data Analysis

- ดูแนวโน้มค่าเฉลี่ยคะแนนรวมถ่วงน้ำหนักที่คิดจาก GAT 20%, PAT1 20% และ PAT3 60% ของปี 55-63 (ไม่เอาปี 53-54 เพราะใช้คนละเกณฑ์) เทียบคะแนนต่ำสุดที่ติดวิศวะ จุฬา TCAS รอบ 3 (รับตรง)
- วิเคราะห์คะแนนของคนที่ได้ GAT 150up, PAT1 Top 1500 และ PAT3 Top 1000 (สมมติฐานคะแนนของกลุ่มคนที่มีลุ้นสอบติดวิศวะ จุฬา)

In [6]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

แนวโน้มค่าเฉลี่ยคะแนน

In [7]:
```
plt.plot(y.index, y.CU_R3_Min, y.index, X.Total_avg)
plt.errorbar(y.index, X.Total_avg, yerr=X.Total_std, fmt='r^')
plt.xlabel('Year')
plt.ylabel('Scores')
plt.legend(['Minimum Score','Combined Average Score'])
plt.title('Minimum admitted scores and combined average scores')
plt.show()
```



เราจะเห็นว่า ค่าเฉลี่ยของคะแนน ไม่ใช่ตัวทำนายที่ดีนัก เพราะแนวโน้มไม่ได้สอดคล้องกับคะแนนต่ำสุดที่ติดวิศวะจุฬาจริง และยังมีส่วนเบี่ยงเบนมาตรฐานที่ใหญ่มากอีกด้วย

แนวโน้มคะแนนของกลุ่มคะแนนมีโอกาสสอบติดจริง

สมมติฐาน: น้องที่จะสอบติดวิศวะจุฬา คือกลุ่มที่ได้คะแนนค่อนข้างสูง กล่าวคือ

- GAT > 150
- PAT1 Top 1500 และ Top 2000 (อาจมีหลายคน ไปยื่นหมอ)
- PAT3 Top 1000 และ Top 800

ดังนั้น เราจะเอาค่าสถิติของคนกลุ่มนี้ มาวิเคราะห์ดูแนวโน้ม

In [8]:
```
MAX_SCORE = 300
SCORE_BIN = 30

def get_top(X_range, exam_name, top_N=1000):
    ''' A method to get the (approximate) score of top_N'th student
    INPUT : X_range (DataFrame) - Score range dataframe of a single year
            exam_name (str) - name of interested exam
            top_N - the interested rank
    OUTPUT: top_N_score (float) - the score of rank top_N
    '''
    current_rank = 0
    current_range = MAX_SCORE - SCORE_BIN
    while True:
        N_stu = X_range[exam_name+'_'+str(current_range)] #Number of students in this score range
        if (current_rank + N_stu) < top_N:
            current_rank += N_stu
            current_range -= SCORE_BIN
        else:
            top_N_score = current_range + SCORE_BIN*((current_rank + N_stu) - top_N)/N_stu
            return top_N_score
```

In [9]:
```
X['PAT1_top1500'] = X.apply(lambda row: get_top(row, 'PAT1', top_N=1500), axis=1)
X['PAT3_top1000'] = X.apply(lambda row: get_top(row, 'PAT3', top_N=1000), axis=1)
```

In [10]:
```
X['PAT1_top2000'] = X.apply(lambda row: get_top(row, 'PAT1', top_N=2000), axis=1)
X['PAT3_top800'] = X.apply(lambda row: get_top(row, 'PAT3', top_N=800), axis=1)
```

In [11]:
```
X.head()
```

Out[11]:

Year	Round	Quota	GAT_N	PAT1_N	PAT3_N	GAT_avg	GAT_std	PAT1_avg	PAT1_std	PAT3_avg	...
2555	1	800	284735	200693	39488	130.59	68.04	39.64	20.07	83.45	...
2556	1	800	323876	243834	51235	114.30	60.63	40.61	20.96	91.11	...
2557	1	600	237419	173708	35302	143.58	61.60	57.40	24.13	99.40	...
2558	1	610	316791	239345	49776	131.97	63.81	51.56	18.85	93.41	...
2559	1	610	324856	253213	57879	115.66	58.73	52.61	21.64	80.66	...

5 rows × 38 columns

In [12]:
```
def get_weighted_avg(X_range, exam_name, limit_score=150):
    ''' A method to get the weighted average score given a min score
    INPUT : X_range (DataFrame) - Score range dataframe of a single year
            exam_name (str) - - name of interested exam
            min_score (int) - the lower bound of interested range, MUST be in [90,120,150,180,210,240,270]
    OUTPUT: weighted_score (float) - the weighted score above the limit_score
            total_students (int) - number of students above the limit_score
    '''
    score_ranges = np.arange(limit_score, MAX_SCORE, SCORE_BIN).astype(int)
    grand_total = 0
    total_students = 0
    for current_range in score_ranges:
        grand_total += X_range[exam_name+'_'+str(current_range)]*(current_range + 0.5*SCORE_BIN)
        total_students += X_range[exam_name+'_'+str(current_range)]

    weighted_score = grand_total/total_students
    return weighted_score, int(total_students)
```

In [13]:
```
X['GAT_150up_avg'], X['GAT_150up_N'] = zip(*X.apply(lambda row: get_weighted_avg(row, 'GAT', limit_score=150), axis=1))
X['PAT1_120up_avg'], X['PAT1_120up_N'] = zip(*X.apply(lambda row: get_weighted_avg(row, 'PAT1', limit_score=120), axis=1))
X['PAT3_150up_avg'], X['PAT3_150up_N'] = zip(*X.apply(lambda row: get_weighted_avg(row, 'PAT3', limit_score=150), axis=1))
```

In [14]:
```
X
```

Out[14]:

Year	Round	Quota	GAT_N	PAT1_N	PAT3_N	GAT_avg	GAT_std	PAT1_avg	PAT1_std	PAT3_avg	...
2555	1	800	284735	200693	39488	130.59	68.04	39.64	20.07	83.45	...
2556	1	800	323876	243834	51235	114.30	60.63	40.61	20.96	91.11	...
2557	1	600	237419	173708	35302	143.58	61.60	57.40	24.13	99.40	...
2558	1	610	316791	239345	49776	131.97	63.81	51.56	18.85	93.41	...
2559	1	610	324856	253213	57879	115.66	58.73	52.61	21.64	80.66	...
2560	1	590	289866	228522	50623	123.93	58.48	42.82	25.34	95.91	...
2561	1	610	230566	177835	37347	144.65	67.82	48.45	26.13	92.32	...
2562	1	430	215585	163251	34493	144.06	66.90	49.05	29.68	90.81	...
2563	1	400	202341	150694	33147	147.50	63.43	62.90	38.08	98.21	...

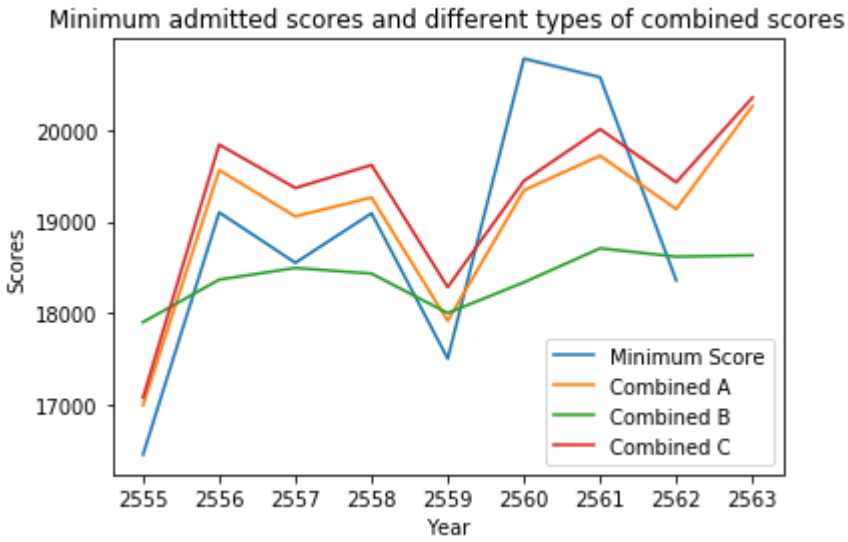
9 rows × 44 columns

เราจะดูคะแนนรวม 2 แบบ ว่าแบบไหนสอดคล้อง Trend คะแนนต่ำสุดที่ติดมากกว่า

- แบบ A: (20% GAT150up.avg + 20% PAT1top1500 + 60% PAT3top1000) * 100 (เต็ม 30000)
- แบบ B: (20% GAT150up.avg + 20% PAT1.120up.avg + 60% PAT3.150up.avg) * 100 (เต็ม 30000)
- แบบ C: (20% GAT150up.avg + 20% PAT1top2000 + 60% PAT3top800) * 100 (เต็ม 30000)

In [15]:
```
X['Combined_A'] = (0.2*X['GAT_150up_avg'] + 0.2*X['PAT1_top1500'] + 0.6*X['PAT3_top1000'])*100
X['Combined_B'] = (0.2*X['GAT_150up_avg'] + 0.2*X['PAT1_120up_avg'] + 0.6*X['PAT3_150up_avg'])*100
X['Combined_C'] = (0.2*X['GAT_150up_avg'] + 0.2*X['PAT1_top2000'] + 0.6*X['PAT3_top800'])*100
```

In [16]:
```
plt.plot(X.index, y.CU_R3_Min, X.index, X.Combined_A, X.index, X.Combined_B, X.index, X.Combined_C)
plt.xlabel('Year')
plt.ylabel('Scores')
plt.legend(['Minimum Score','Combined A','Combined B','Combined C'])
plt.title('Minimum admitted scores and different types of combined scores')
plt.show()
```



ข้อสังเกต

- การคิดด้วยค่าเฉลี่ยถ่วงน้ำหนักอย่างเดียว (แบบ B) แทบใช้ในการทำนายไม่ได้เลย
- การคิดแบบ A และ C ใช้ได้เกือบทุกปี ยกเว้นปี 2560 ที่มีปัญหา อาจเป็นเพราะปี 2560 มีคนคะแนนสูงมายื่นสมัครรับตรงวิศวะจุฬาเยอะ (แต่อาจมีสละสิทธิ์ทีหลัง)

ด้วยเหตุนี้ เราจึงต้องลองให้ Learning Algorithms ช่วยหาโมเดลการตัดสินใจให้ เพราะ Exploratory Data Analysis ทำนายปี 2560 ยังไม่ดีนัก

In [17]:
```
X.columns
```

Out[17]:
```
Index(['Round', 'Quota', 'GAT_N', 'PAT1_N', 'PAT3_N', 'GAT_avg', 'GAT_std',
       'PAT1_avg', 'PAT1_std', 'PAT3_avg', 'PAT3_std', 'Total_avg',
       'Total_std', 'GAT_90', 'GAT_120', 'GAT_150', 'GAT_180', 'GAT_210',
       'GAT_240', 'GAT_270', 'PAT1_90', 'PAT1_120', 'PAT1_150', 'PAT1_180',
       'PAT1_210', 'PAT1_240', 'PAT1_270', 'PAT3_90', 'PAT3_120', 'PAT3_150',
       'PAT3_180', 'PAT3_210', 'PAT3_240', 'PAT3_270', 'PAT1_top1500',
       'PAT3_top1000', 'PAT1_top2000', 'PAT3_top800', 'GAT_150up_avg',
       'GAT_150up_N', 'PAT1_120up_avg', 'PAT1_120up_N', 'PAT3_150up_avg',
       'PAT3_150up_N', 'Combined_A', 'Combined_B', 'Combined_C'],
      dtype='object')
```

Data Preposessing and Feature Engineering

ในส่วนนี้ เราจะมาเตรียม data ให้พร้อมสำหรับโมเดล Machine Learning และคิด Feature ที่คาดว่าจะทำให้ทำนายผลได้แม่นยำขึ้น

รายการ Features

- `Quota` จำนวนรับ
- `GAT_150` , `GAT_180` ,..., `GAT_270` จำนวนคนที่ได้คะแนน GAT ในช่วงต่างๆ ตั้งแต่ 150 คะแนนขึ้นไป
- `PAT1_120` , `PAT1_150` ,..., `PAT1_270` จำนวนคนที่ได้คะแนน PAT1 ในช่วงต่างๆ ตั้งแต่ 120 คะแนนขึ้นไป
- `PAT3_150` , `PAT3_180` ,..., `PAT3_270` จำนวนคนที่ได้คะแนน PAT3 ในช่วงต่างๆ ตั้งแต่ 150 คะแนนขึ้นไป
- `GAT_150up_avg` คะแนนเฉลี่ยถ่วงน้ำหนักของคนที่ได้ GAT 150up
- `PAT1_top1500` คะแนน PAT1 อันดับที่ 1500
- `PAT3_top1000` คะแนน PAT3 อันดับที่ 1000

In [18]:
```
selected_features = ['Quota', 'GAT_120', 'GAT_150', 'GAT_180', 'GAT_210', 'GAT_240',
'GAT_270',
                     'PAT1_120', 'PAT1_150', 'PAT1_180','PAT1_210', 'PAT1_240', 'PAT1
_270',
                     'PAT3_150', 'PAT3_180', 'PAT3_210', 'PAT3_240', 'PAT3_270',
                     'PAT1_top1500', 'PAT3_top1000', 'GAT_150up_avg']
```

In [19]:
```
X_selected = X[selected_features]
```

In [20]:
```
X_selected
```

Out[20]:

Year	Quota	GAT_120	GAT_150	GAT_180	GAT_210	GAT_240	GAT_270	PAT1_120	PAT1_150	PAT1_18
2555	800	32588	38943	43860	23222	11365	3499	1224	546	23
2556	800	42493	36997	88012	16018	7817	2067	1567	711	31
2557	600	32894	44766	42862	22601	91890	1524	2681	1057	38
2558	610	40129	39092	46904	28061	10374	2630	1160	401	15
2559	610	53343	40077	33783	15417	4671	592	2206	938	38
2560	590	44402	49806	36845	15734	3933	442	2005	1058	59
2561	610	24289	43416	45798	24062	10943	2349	2163	1038	51
2562	430	28503	30485	31003	21141	14017	4822	2798	1493	75
2563	400	28906	38179	34084	20840	10809	2347	4187	2537	156

9 rows × 21 columns

In [21]:
```
X_standardized = (X_selected - X_selected.mean())/(X_selected.std())
```

In [22]:
```
X_standardized
```

Out[22]:

Year	Quota	GAT_120	GAT_150	GAT_180	GAT_210	GAT_240	GAT_270	PAT1_120	PAT1_150	P
2555	1.424672	-0.408602	-0.231118	-0.054329	0.561823	-0.254465	0.912881	-1.065514	-0.854916	-(
2556	1.424672	0.654741	-0.590157	2.512395	-1.101328	-0.382358	-0.135805	-0.699025	-0.593960	-(
2557	-0.040705	-0.375751	0.843230	-0.112347	0.418456	2.648185	-0.533456	0.491264	-0.046744	-(
2558	0.032564	0.400956	-0.203627	0.122630	1.678978	-0.290187	0.276492	-1.133897	-1.084241	-(
2559	0.032564	1.819532	-0.021894	-0.640144	-1.240078	-0.495760	-1.215981	-0.016265	-0.234948	-(
2560	-0.113974	0.859679	1.773114	-0.462138	-1.166894	-0.522363	-1.325829	-0.231030	-0.045162	(
2561	0.032564	-1.299533	0.594154	0.058334	0.755749	-0.269676	0.070710	-0.062209	-0.076793	-(
2562	-1.286276	-0.847143	-1.791626	-0.801756	0.081393	-0.158869	1.881743	0.616277	0.642813	(
2563	-1.506082	-0.803879	-0.372077	-0.622646	0.011902	-0.274507	0.069245	2.100399	2.293953	2

9 rows × 21 columns

In [23]:
```
X_train = X_standardized.drop([2563])
y_train = y.drop([2563])

X_test = X_standardized.loc[[2563]]
```

In [24]:
```
X_train
```

Out[24]:

Year	Quota	GAT_120	GAT_150	GAT_180	GAT_210	GAT_240	GAT_270	PAT1_120	PAT1_150	P
2555	1.424672	-0.408602	-0.231118	-0.054329	0.561823	-0.254465	0.912881	-1.065514	-0.854916	-(
2556	1.424672	0.654741	-0.590157	2.512395	-1.101328	-0.382358	-0.135805	-0.699025	-0.593960	-(
2557	-0.040705	-0.375751	0.843230	-0.112347	0.418456	2.648185	-0.533456	0.491264	-0.046744	-(
2558	0.032564	0.400956	-0.203627	0.122630	1.678978	-0.290187	0.276492	-1.133897	-1.084241	-(
2559	0.032564	1.819532	-0.021894	-0.640144	-1.240078	-0.495760	-1.215981	-0.016265	-0.234948	-(
2560	-0.113974	0.859679	1.773114	-0.462138	-1.166894	-0.522363	-1.325829	-0.231030	-0.045162	(
2561	0.032564	-1.299533	0.594154	0.058334	0.755749	-0.269676	0.070710	-0.062209	-0.076793	-(
2562	-1.286276	-0.847143	-1.791626	-0.801756	0.081393	-0.158869	1.881743	0.616277	0.642813	(

8 rows × 21 columns

In [25]:
```
y_train
```

Out[25]:

Year	CU_R3_Min
2555	16465.0
2556	19100.0
2557	18552.0
2558	19090.0
2559	17510.0
2560	20773.0
2561	20570.0
2562	18360.0

Model Selection and Training

ต่อจากนี้ เราจะเริ่มทำการหาโมเดล Machine Learning ที่จะทำนายคะแนนต่ำสุดที่ติดวิศวะ จุฬา รอบ 3 โดยเบื้องต้น จะเน้นไปที่โมเดล Regression ดังนี้

- Linear Regression
- Random Forest Regressor
- AdaBoost Regressor

In [26]:
```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor
from sklearn.metrics import mean_absolute_error, make_scorer
from sklearn.model_selection import GridSearchCV, cross_val_score, cross_validate
```

In [27]:
```
RF = RandomForestRegressor(random_state=42)
ADA = AdaBoostRegressor(random_state=42)
LR = LinearRegression()

models = [RF, ADA, LR]
model_names = ['Random Forest', 'AdaBoost', 'Linear Regression']
```

In [28]:
```
# Calculate the scores for all models using cross-validation method
n_models = len(models)
for i in range(0, n_models):
    model = models[i]
    model_name = model_names[i]
    scores = cross_val_score(model, X_train, y_train.values.ravel(), cv=3, scoring='n
eg_mean_absolute_error')
    mae = - scores.mean()
    print('The mean mae for {:<30} is {:>10.3f}'.format(model_name, mae))
```

```
The mean mae for Random Forest                 is    951.895
The mean mae for AdaBoost                      is   1260.667
The mean mae for Linear Regression             is    468.786
```

พบว่า โมเดล Linear Regression แม่นยำสุด ดังนั้น เราจะลอง Train โมเดล Linear Regression ก่อน

Linear Regression

In [29]:
```
LR_CV_results = cross_validate(LR, X_train, y_train, cv=3, scoring='neg_mean_absolute
_error',
                               return_estimator=True)
```

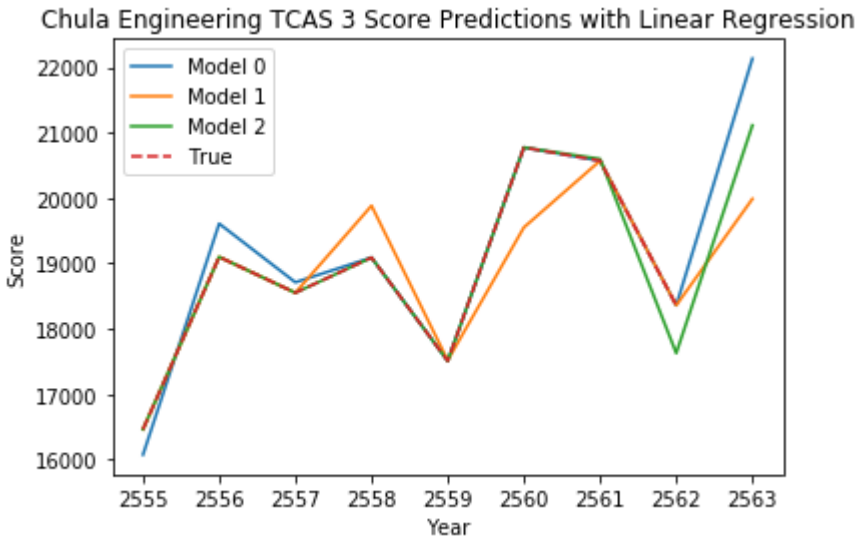
In [30]:
```
LR_CV_results
```

Out[30]:
```
{'fit_time': array([0.00299072, 0.00199461, 0.00203776]),
 'score_time': array([0.0010004 , 0.00099778, 0.00095415]),
 'estimator': (LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normal
ize=False),
  LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False),
  LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)),
 'test_score': array([-352.25653978, -674.76791923, -379.33339246])}
```

In [31]:
```
LR_model_CV = LR_CV_results['estimator']
```

In [32]:
```
LR_predictions = []
for estimator in LR_model_CV:
    train_preds = estimator.predict(X_train)
    test_preds = estimator.predict(X_test)
    LR_predictions.append(np.append(train_preds,test_preds))
```

In [33]:
```
for i in range(len(LR_predictions)):
    plt.plot(np.arange(2555,2564), LR_predictions[i])
plt.plot(np.arange(2555,2563), y_train.values.ravel(), linestyle='dashed')
plt.xticks(np.arange(2555,2564))
plt.legend(['Model {}'.format(i) for i in range(len(LR_predictions))] + ['True'])
plt.xlabel('Year')
plt.ylabel('Score')
plt.title('Chula Engineering TCAS 3 Score Predictions with Linear Regression')
plt.show()
```



Model 0 มีคะแนนที่ดีที่สุด และสอดคล้องกับคะแนนต่ำสุดย้อนหลังช่วงปี 2558-2562 มากที่สุด

In [34]:
```
print("คะแนนทำนายต่ำสุด วิศวะ จุฬา = {:.2f}".format(LR_predictions[0][-1]))
```

```
คะแนนทำนายต่ำสุด วิศวะ จุฬา = 22131.87
```

แต่ทั้งนี้ Model 0 เคยมีประวัติทำนายคะแนนสูงกว่าความเป็นจริงในปี 2556 และ 2557 และจะเห็นว่า Model 2 นั้นทำนายปี 2555-2561 ได้แม่นยำกว่า จะมีปัญหาแค่ปี 2562 เพื่อให้การทำนาย ไม่ "มั่นใจ" จนเกินไป เราจะรายงานผลดังต่อไปนี้

> คะแนน Model 2 ± (คะแนนโมเดล 0 - คะแนนโมเดล 1)/2

In [35]:
```
model0_score = LR_predictions[0][-1]
model1_score = LR_predictions[1][-1]
model2_score = LR_predictions[2][-1]
confidence_int = (model0_score - model1_score)/2

print("คะแนนทำนายต่ำสุด วิศวะ จุฬา = {:.2f} ± {:.2f}".format(model2_score, confidence_in
t))
```

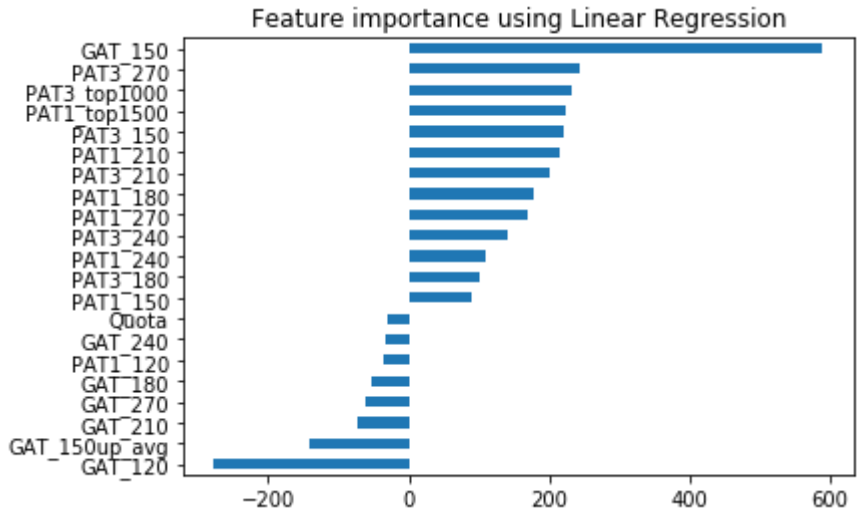
```
คะแนนทำนายต่ำสุด วิศวะ จุฬา = 21111.63 ± 1072.08
```

ลองดูว่า สัมประสิทธิ์ของตัวแปรไหนบ้างที่มีความสำคัญต่อการทำนายคะแนน

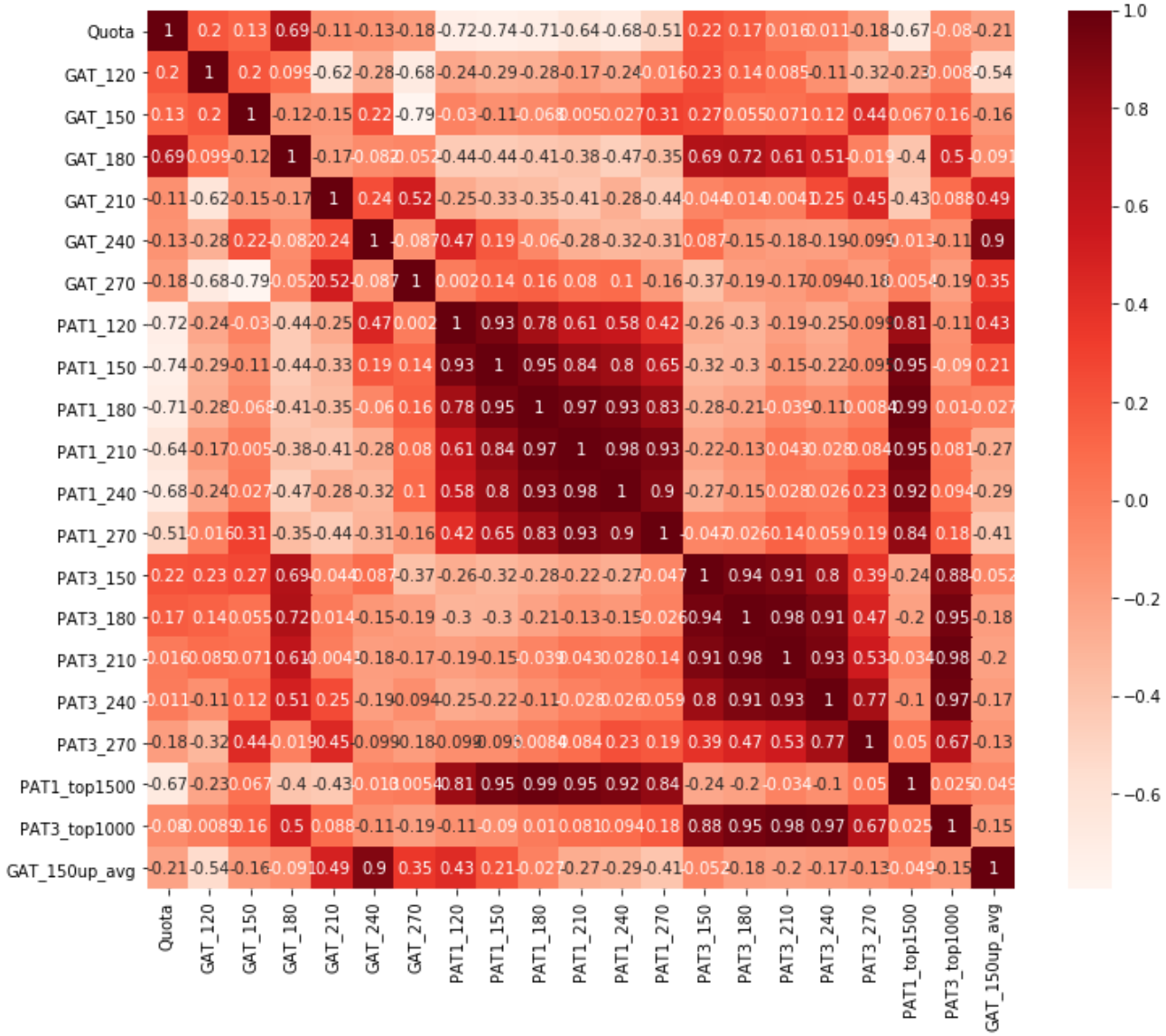
In [36]:
```
coefficients = pd.Series(LR_model_CV[2].coef_[0], index=X_train.columns)
```

In [37]:
```
imp_coef = coefficients.sort_values()
imp_coef.plot(kind = "barh")
plt.title("Feature importance using Linear Regression")
```

Out[37]:
```
Text(0.5, 1.0, 'Feature importance using Linear Regression')
```



In [38]:
```
# Pearson Correlation
plt.figure(figsize=(12,10))
cor = X_train.corr()
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()
```



ข้อสังเกต

- คะแนน PAT3 ตามมาด้วย PAT1 กลุ่มที่ได้คะแนนสูง มีผลมากๆ ในการทำนาย
- น่าสนใจว่าจำนวนคนที่ได้คะแนน GAT ในช่วง 150-179 มีผลมากที่สุดต่อคะแนนทำนาย ซึ่งจาก Correlation Plot เราพบว่า Feature นี้ Correlate กับ PAT3_270 ค่อนข้างมาก (corr = 0.44) อาจคาดเดาได้ว่า ถ้าปีนั้นมีคนได้คะแนน GAT ช่วง 150-179 เยอะ จะทำให้มีคนได้คะแนน PAT3 สูงๆ จำนวนเยอะขึ้น แต่นี่เป็นเพียงการคาดเดาเท่านั้น

Random Forest Regressor

In [39]:
```
def get_best_model(model, parameters):
    ''' Return the best tuned model with its mse score
    INPUT: model - untuned model
    parameters - dictionary of tuning hyper parameters
    OUTPUT: best_model - the best tuned model
    mse - model cv mse
    '''
    scorer = make_scorer(mean_absolute_error, greater_is_better=False)
    grid_obj = GridSearchCV(model, parameters, scoring = scorer, cv=3)
    grid_fit = grid_obj.fit(X_train, y_train.values.ravel())
    best_model = grid_fit.best_estimator_
    mae = - grid_fit.best_score_

    return best_model, mae
```

In [40]:
```
parameters = {"bootstrap" : [True],
              "max_depth" : [2,5,7,10],
              "min_samples_leaf" : [1],
              "min_samples_split" : [2],
              "n_estimators": [200, 300, 400, 500, 700]}

RF_best_model, RF_best_mae = get_best_model(RF, parameters)
print(RF_best_model)
print("The mae for RF best model on training set is: {:.3f}".format(RF_best_mae))
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=2, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=500, n_jobs=None, oob_score=False,
                      random_state=42, verbose=0, warm_start=False)
The mae for RF best model on training set is: 923.244
```

In [41]:
```
RF_CV_results = cross_validate(RF_best_model, X_train, y_train.values.ravel(), cv=3,
scoring='neg_mean_absolute_error',
                               return_estimator=True)
```

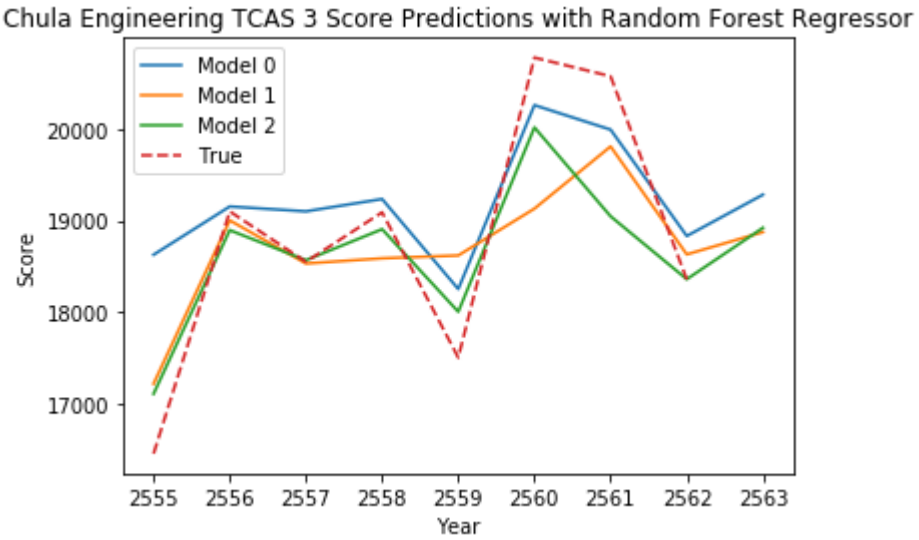
In [42]:
```
RF_CV_results
```

Out[42]:
```
{'fit_time': array([0.46285844, 0.4318459 , 0.44281602]),
 'score_time': array([0.02393651, 0.02393508, 0.02194452]),
 'estimator': (RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                           max_depth=2, max_features='auto', max_leaf_nodes=None,
                           max_samples=None, min_impurity_decrease=0.0,
                           min_impurity_split=None, min_samples_leaf=1,
                           min_samples_split=2, min_weight_fraction_leaf=0.0,
                           n_estimators=500, n_jobs=None, oob_score=False,
                           random_state=42, verbose=0, warm_start=False),
  RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                           max_depth=2, max_features='auto', max_leaf_nodes=None,
                           max_samples=None, min_impurity_decrease=0.0,
                           min_impurity_split=None, min_samples_leaf=1,
                           min_samples_split=2, min_weight_fraction_leaf=0.0,
                           n_estimators=500, n_jobs=None, oob_score=False,
                           random_state=42, verbose=0, warm_start=False),
  RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                           max_depth=2, max_features='auto', max_leaf_nodes=None,
                           max_samples=None, min_impurity_decrease=0.0,
                           min_impurity_split=None, min_samples_leaf=1,
                           min_samples_split=2, min_weight_fraction_leaf=0.0,
                           n_estimators=500, n_jobs=None, oob_score=False,
                           random_state=42, verbose=0, warm_start=False)),
 'test_score': array([ -922.02266667, -1084.64666667,  -763.06166667])}
```

In [43]:
```
RF_model_CV = RF_CV_results['estimator']
```

In [44]:
```
RF_predictions = []
for estimator in RF_model_CV:
    train_preds = estimator.predict(X_train)
    test_preds = estimator.predict(X_test)
    RF_predictions.append(np.append(train_preds,test_preds))
```

In [45]:
```
for i in range(len(RF_predictions)):
    plt.plot(np.arange(2555,2564), RF_predictions[i])
plt.plot(np.arange(2555,2563), y_train.values.ravel(), linestyle='dashed')
plt.xticks(np.arange(2555,2564))
plt.legend(['Model {}'.format(i) for i in range(len(RF_predictions))] + ['True'])
plt.xlabel('Year')
plt.ylabel('Score')
plt.title('Chula Engineering TCAS 3 Score Predictions with Random Forest Regressor')
plt.show()
```

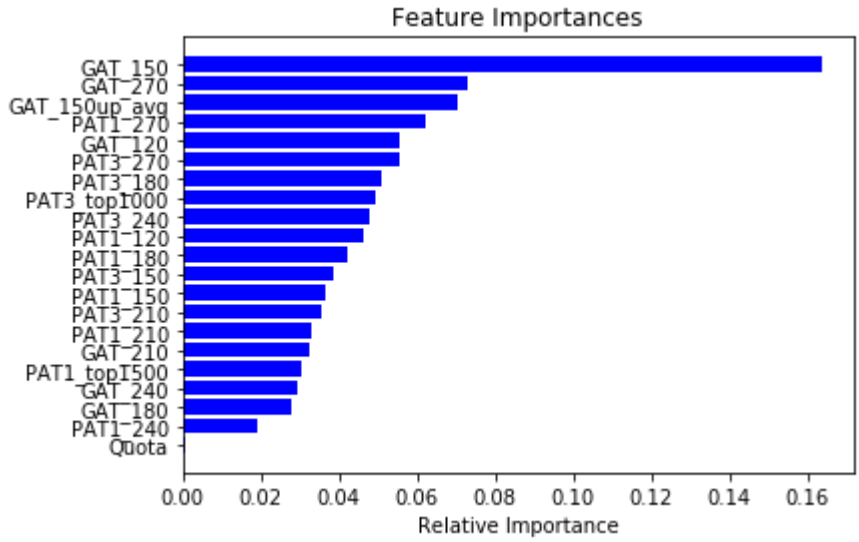


จะเห็นชัดเจนว่า Random Forest ทำนายสู้ Linear Regression ไม่ได้ในที่นี้

ลองดูความสำคัญของ Features ต่างๆ ของโมเดล 0 และ 2 ที่ Error น้อย

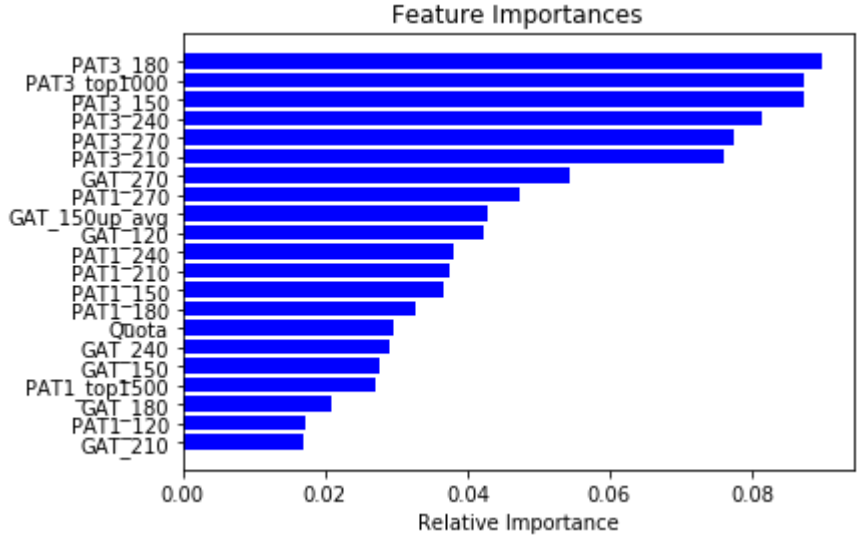
In [46]:
```
features = X_train.columns
importances = RF_model_CV[0].feature_importances_
indices = np.argsort(importances)

plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



In [47]:
```
features = X_train.columns
importances = RF_model_CV[2].feature_importances_
indices = np.argsort(importances)

plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```



สิ่งที่อาจทำเพิ่ม ถ้ามีเวลามากขึ้น

- เลือก Features ใหม่ๆ มาวิเคราะห์ อาจทำให้ทำนายคะแนนได้แม่นยำขึ้น
- ลองโมเดลอื่นๆ ดู เช่น อาจลอง Train Neural Network ดู 555
- ลองทำนายคะแนนของสถาบันอื่น

สรุปผลการวิเคราะห์

จากการทำนายด้วยโมเดล Linear Regression พบว่า คะแนนทำนายต่ำสุดเพื่อสอบติด วิศวะ จุฬ้า TCAS รอบ 3 ปี 2563 โดยใช้ข้อมูลปี 2555-2562 คือ

> **21,112 ± 1,072 คะแนน**

ข้อสรุปจากการวิเคราะห์

- คะแนนปีนี้ มีแนวโน้ม **เฟ้อ**มากกว่าทุกปีที่ผ่านมา (สังเกตเบื้องต้นได้จากจำนวนคนทำ PAT1 ได้เกิน 120 ที่เพิ่มขึ้นจากปีก่อนๆ ถึงเท่าตัว)
- คะแนนปีนี้ มีแนวโน้มใกล้เคียงคะแนนปี **2560** และ **2561**
- น้องที่มีคะแนนในช่วง 20,000 มีสิทธิ์ลุ้นสอบติดนะครับ (แม้แต่หากน้องได้ 19,XXX ก็ควรลองยื่นเป็นอันดับ 1 ดูครับ)
- แต่พี่ขอย้ำอีกทีนะครับว่า คะแนนทำนายนี้ เป็นเพียงคะแนนทำนายเท่านั้น ทำให้อาจมีปัจจัยอื่นที่โมเดลไม่ได้คำนึงถึงครับ

ทั้งนี้ การเลือกคณะ พี่แนะนำให้น้องเลือกด้วยหลักการดังนี้นะครับ

- **อันดับ 1:** เลือกตามความอยากเลยโดยไม่ต้องแคร์คะแนนที่น้องได้ จะได้ไม่คาใจครับ
- **อันดับ 2-3:** คะแนนของน้อง อยู่ในช่วงค่าต่ำสุดที่สอบติดของ 3 ปีย้อนหลัง
- **อันดับ 4-5:** คะแนนของน้อง เกินคะแนนต่ำสุดที่สอบติดของ 3 ปี ย้อนหลัง อย่างน้อย 1500 คะแนน (สเกลเต็ม 30000)
- **อันดับ 6:** คะแนนของน้อง เกินคะแนนต่ำสุดที่สอบติดของ 3 ปี ย้อนหลัง อย่างน้อย 3000 คะแนน (สเกลเต็ม 30000)

และน้องสามารถวิเคราะห์คะแนนและดูสถิติเพิ่มเติม ได้ที่แอพพลิเคชั่น TCASter (https://tcaster.net/ (https://tcaster.net/)) หรือติดตามข่าวสารได้ที่ Facebook Fanpage (https://www.facebook.com/TCASterApp/ (https://www.facebook.com/TCASterApp/)) ครับ

แล้วก็ เดี๋ยวพี่จะวิเคราะห์คะแนน กสพท ให้ด้วย รอติดตามได้ทาง Instagram ของพี่ ที่ https://www.instagram.com/pgreatondemand (https://www.instagram.com/pgreatondemand) นะครับ

ขอให้น้องๆ สอบติดได้ตามที่ตั้งใจไว้นะ :-)

- พี่เกรท ออนดีมานด์