# Group 15 is coming

Alessandro Vinci
Rocco Gazzaneo
Yufeng Xing
Irene Tricarico
Samantha Macilwaine

vodafone

# The three-step approach

**What we did**

**How we improved it**

**At the end what does it mean…?**

# What we did

# Dummy Variables

## How

```
def clean(dummy, threshold):
    l = []
    #just to index easily
    dummy = np.array(dummy)
    for i in range(dummy.shape[1]):
        #check how many stores per category
        l.append(np.sum(dummy[:,i]))
    l = np.array(l)
    col_to_keep = []
    #check if the category has more than threshold observations
    for j in range(len(l > threshold)):
        #take the category with highest number of stores
        if (l > threshold)[j] == True:
            #if enough add it into the list
            col_to_keep.append(j)

    return col_to_keep
```

## Why

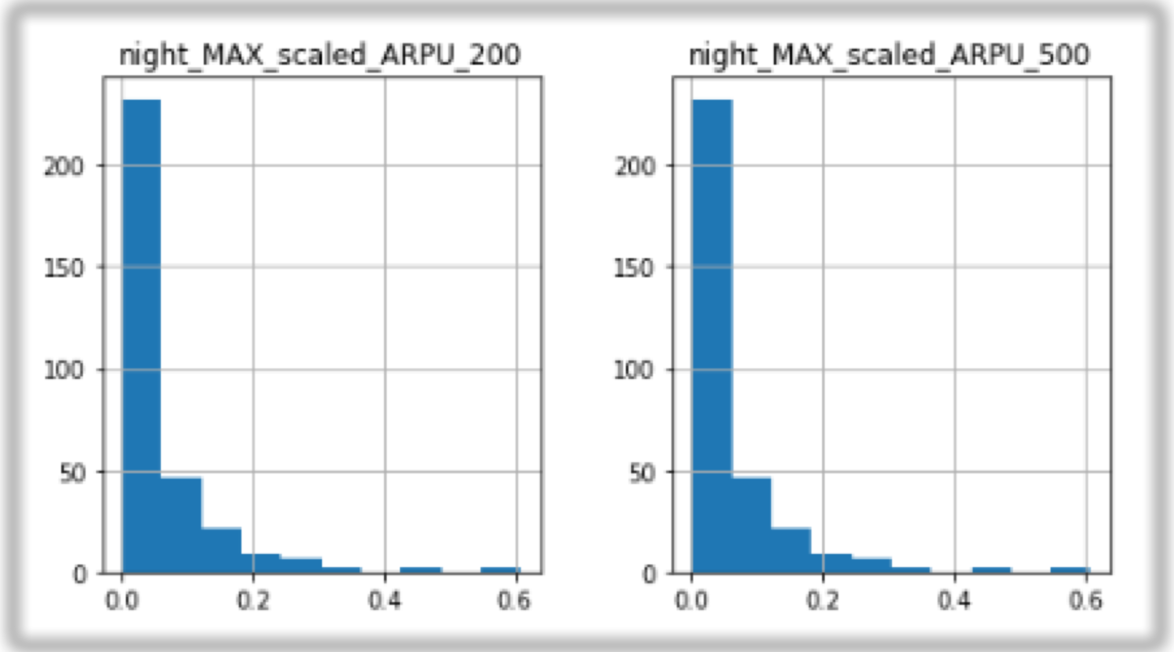
$R^2$

## How

```
AVG1 = (data['night_MIN_scaled_ARPU_200'] + data['night_MAX_scaled_ARPU_200']) / 2
AVG2 = (data['midday_MIN_scaled_ARPU_200'] + data['midday_MAX_scaled_ARPU_200']) / 2
AVG3 = (data['weekend_MIN_scaled_ARPU_200'] + data['weekend_MAX_scaled_ARPU_200']) / 2
AVG4 = (data['night_MIN_scaled_ARPU_500'] + data['night_MAX_scaled_ARPU_500']) / 2
```

# Average
# Min-Max ARPU

## Why

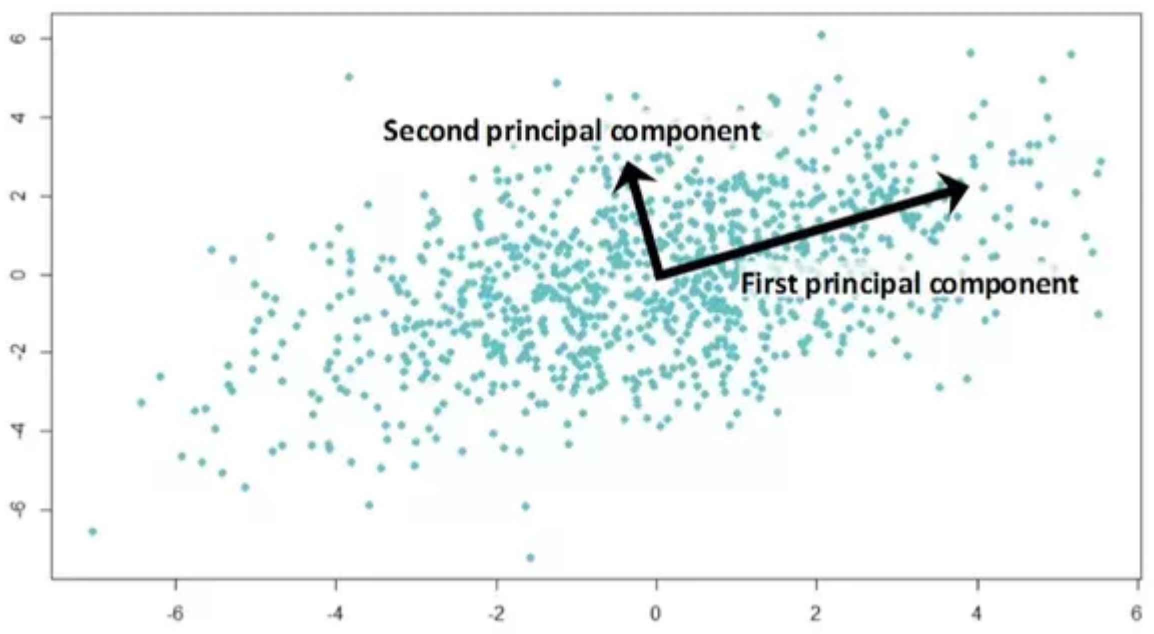


## How

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

pca = PCA()
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)

explained_variance = pca.explained_variance_ratio_
print(np.cumsum(explained_variance))
```

# PCA

## Why

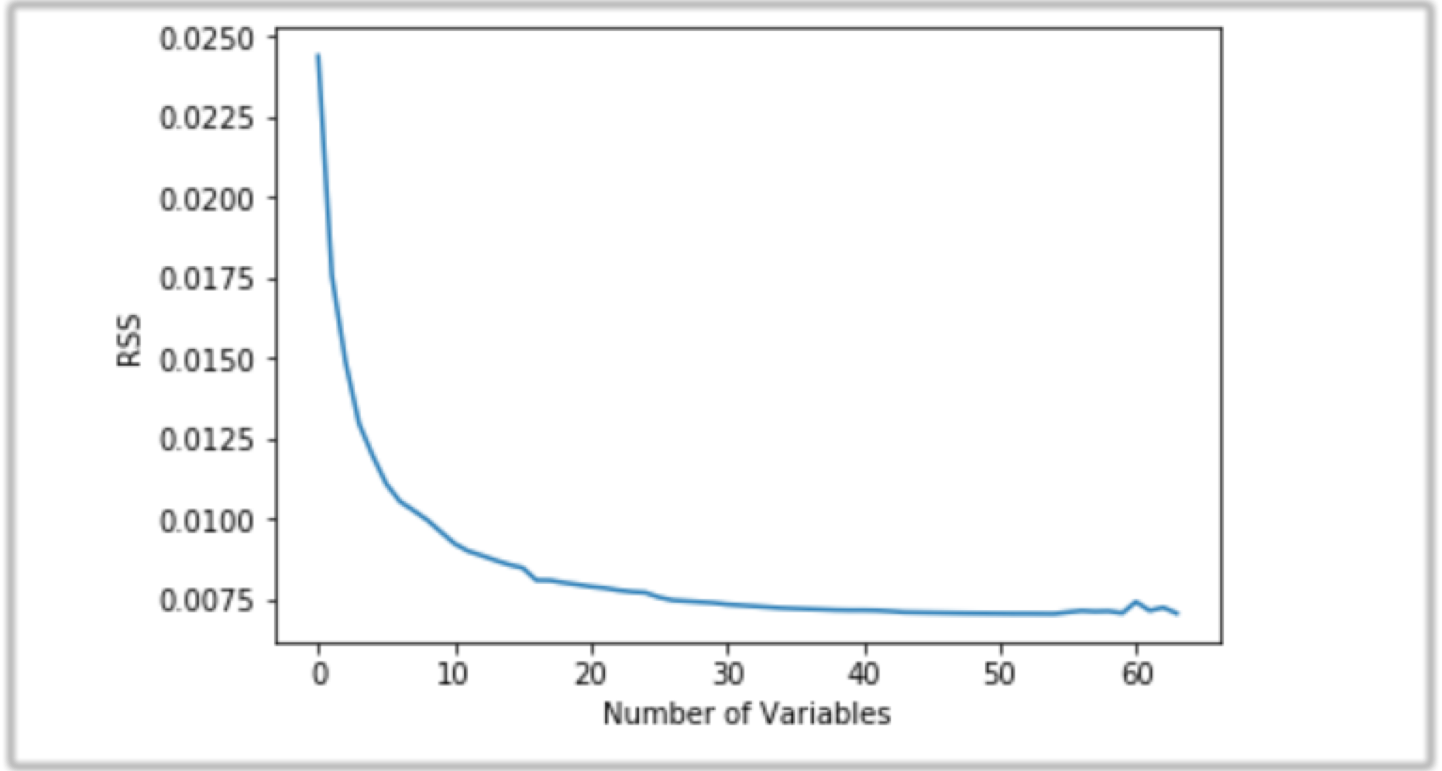


# Forward Selection

## How

```
1) RSS = []
2) dataframe = []
3) for feature in features:
       model.fit(dataframe + feature, y)
       SS = np.sum(y - f(x)**2)
       RSS.append(SS)
4) feature = np.argmin(RSS)
5) dataframe.add(feature)
6) drop feature from features
7) go back to 3
```

## Why



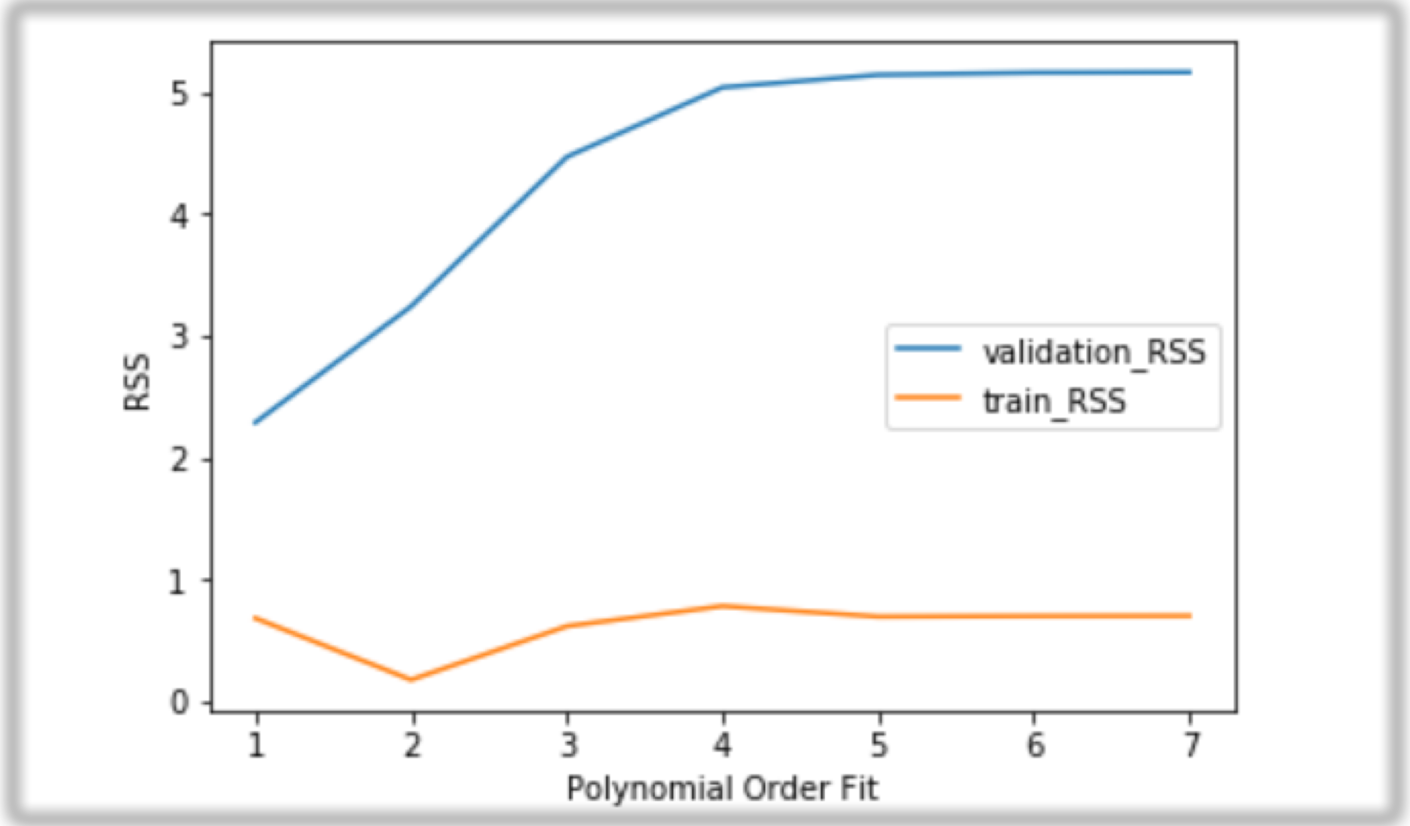
# Linearity

```
y_pred = model.predict(datadf)
y_squared = pd.DataFrame(y_pred**2)
y_third = pd.DataFrame(y_pred**3)

merged_df = pd.concat([datadf, y_squared, y_third], axis = 1)

model = LinearRegression()
model.fit(merged_df, y)
y_merged = model.predict(merged_df)

RSS_restr = np.sum((y_pred - y)**2)
RSS_unr = np.sum((y_merged - y)**2)
j = 2
F_test = ((RSS_restr - RSS_unr) / j) / (RSS_unr / (322 - 6 - 1))

Critical_value = 4.667

print(F_test)
```

```
4.036142316552256
```

# Tests

# Overfitting



# Choose one model and optimize its hyperparameters by Grid search

- Brute-Force Approach
- Try all the possible combinations
- Get the lowest RSS

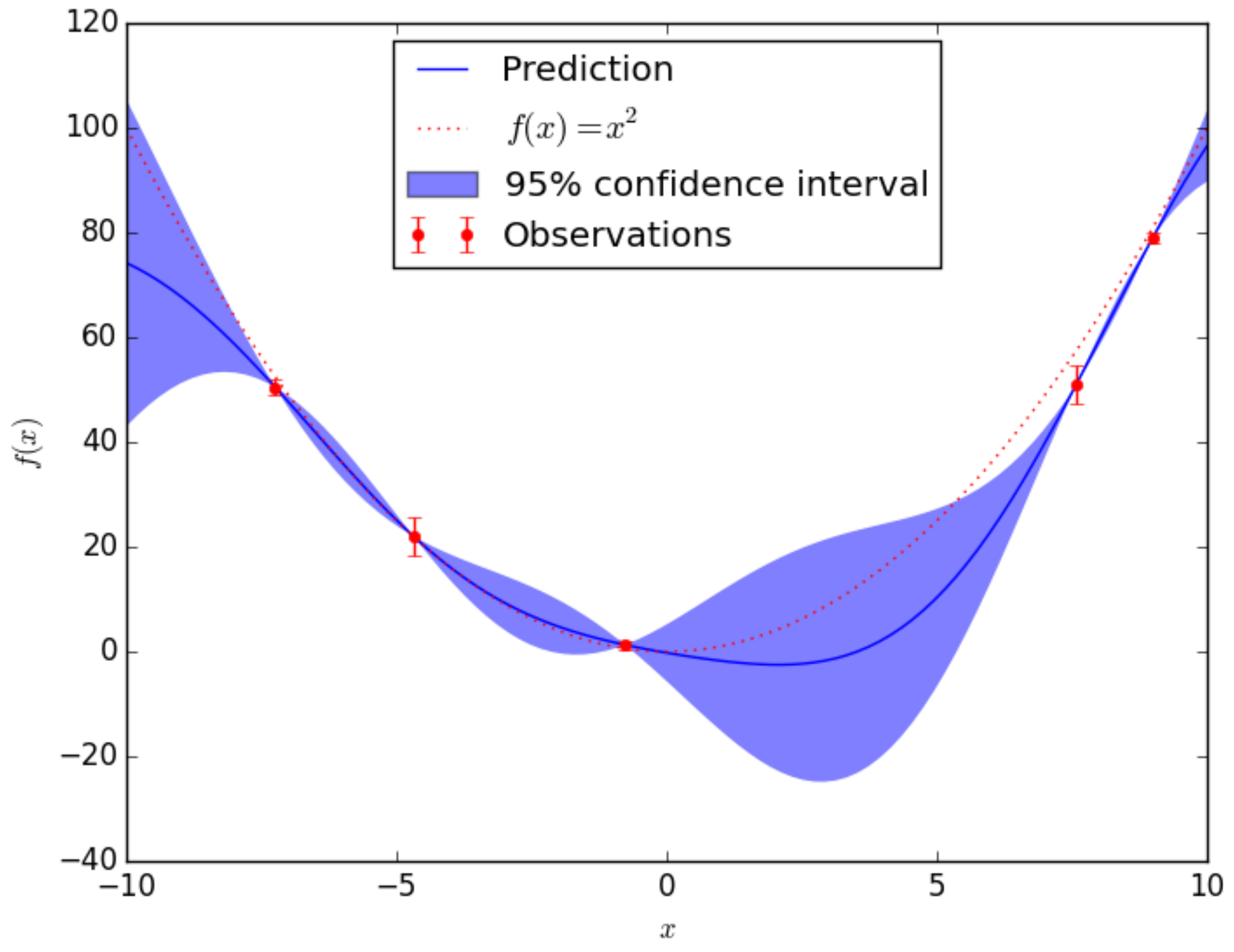
```
C_values = [50, 55, 60, 70,100, 105,110,115,120, 200]
gamma_values = [0.01, 0.03, 0.1, 0.2, 0.11, 0.15, 0.25, 0.3, 0.09, 1, 10, 100]

best_RSS_test = 1000
best_params = {'C': None, 'gamma':None}

for C in C_values:
    for gamma in gamma_values:
        model = SVR(C = C, gamma = gamma, cache_size = 7000)
        model.fit(x_A, y_A)
        predictions = model.predict(x_B)
        RSS_test = np.sum((y_B - predictions)**2) / x_B.shape[0]


        if RSS_test < best_RSS_test:
            best_RSS_test = RSS_test
            best_params['C'] = C
            best_params['gamma'] = gamma
print(f'best RSS test = {best_RSS_test}')
print(best_params)
```

# How we improved it

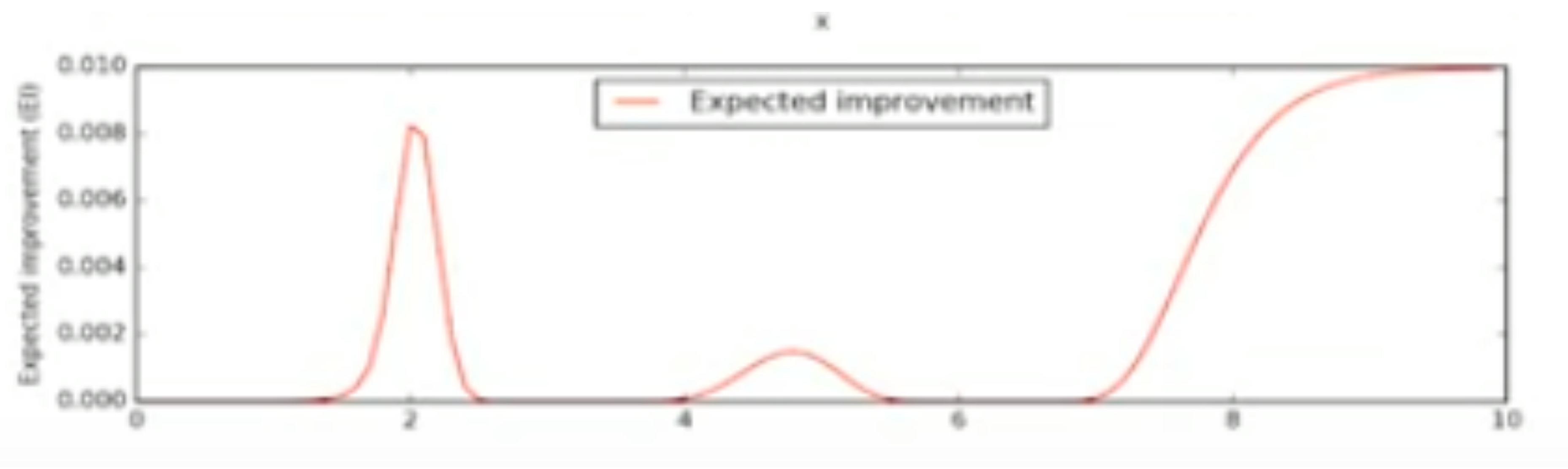
# Bayesian optimization approach

# Loss function modeled with Gaussian Processes

- A Gaussian process generates Functions instead of Random Variables
- Returns mean and Variance of a Normal distributions over all possible values of f at x



# Next Hyperparameter through Expected Improvement Function



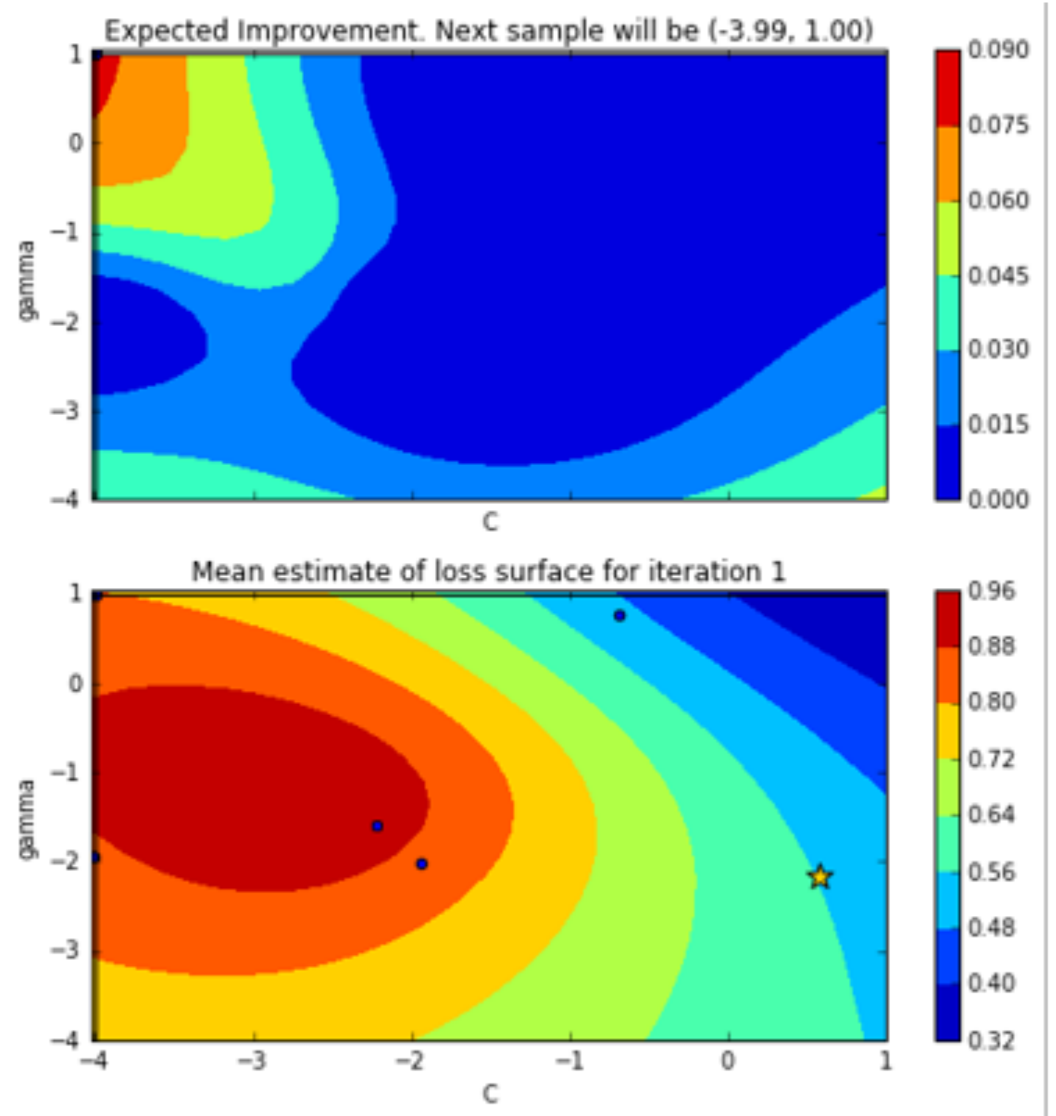

$$EI(\theta) = \mathbb{E}[\max_{\theta}\{0, f_{\mathcal{M}}(\theta) - f_{\mathcal{M}}(\hat{\theta})\}],$$

$$\theta_{new} = \operatorname*{argmax}_{\theta} EI(\theta)$$

# Next Hyperparameter through Expected Improvement Function

$$EI(\theta) = \begin{cases} \left(\mu(\theta) - f(\hat{\theta})\right)\Phi(Z) + \sigma(\theta)\phi(Z), & \sigma(\theta) > 0 \\ 0, & \sigma(\theta) = 0 \end{cases}$$

$$Z = \frac{\mu(\theta) - f(\hat{\theta})}{\sigma(\theta)}$$



# The results

```
params_couples, RSS_min = bayesian_optimisation(n_iters=30,
                                                 sample_loss=sample_loss,
                                                 bounds=bounds,
                                                 n_pre_samples=3,
                                                 random_search=100000)
print(np.argmin(RSS_min))
print(RSS_min[np.argmin(RSS_min)])
best_parameters = params_couples[np.argmin(RSS_min)]
print(best_parameters)
```

```
23
0.009202683468072811
[5.50006865e+01 3.14685669e-02]
```

```
model = SVC(gamma = best_params['gamma'], C = best_params['C'])
model.fit(X_train, y_train)
predicted_classes = model.predict(X_test)

accuracy = accuracy_score(y_test,predicted_classes)
print('accuracy svc= ',accuracy)
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print(confusion_matrix(y_test,predicted_classes))
print(classification_report(y_test,predicted_classes))
```

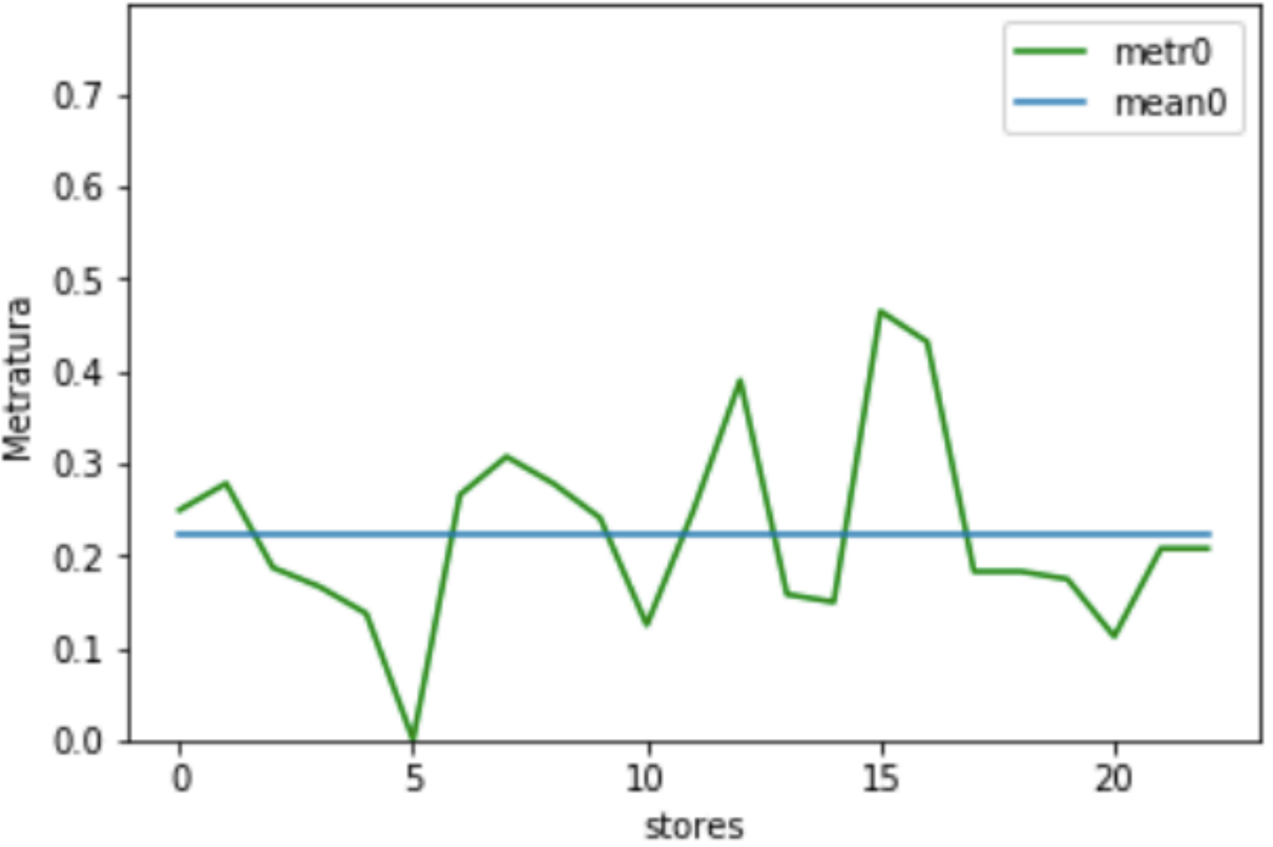
```
accuracy svc=  0.7076923076923077
[[11  6  0]
 [ 7 14  5]
 [ 0  1 21]]
             precision    recall  f1-score   support

          0       0.61      0.65      0.63        17
          1       0.67      0.54      0.60        26
          2       0.81      0.95      0.88        22

avg / total       0.70      0.71      0.70        65
```

At the end what does it mean…

**Results are not the end, but the beginning…**

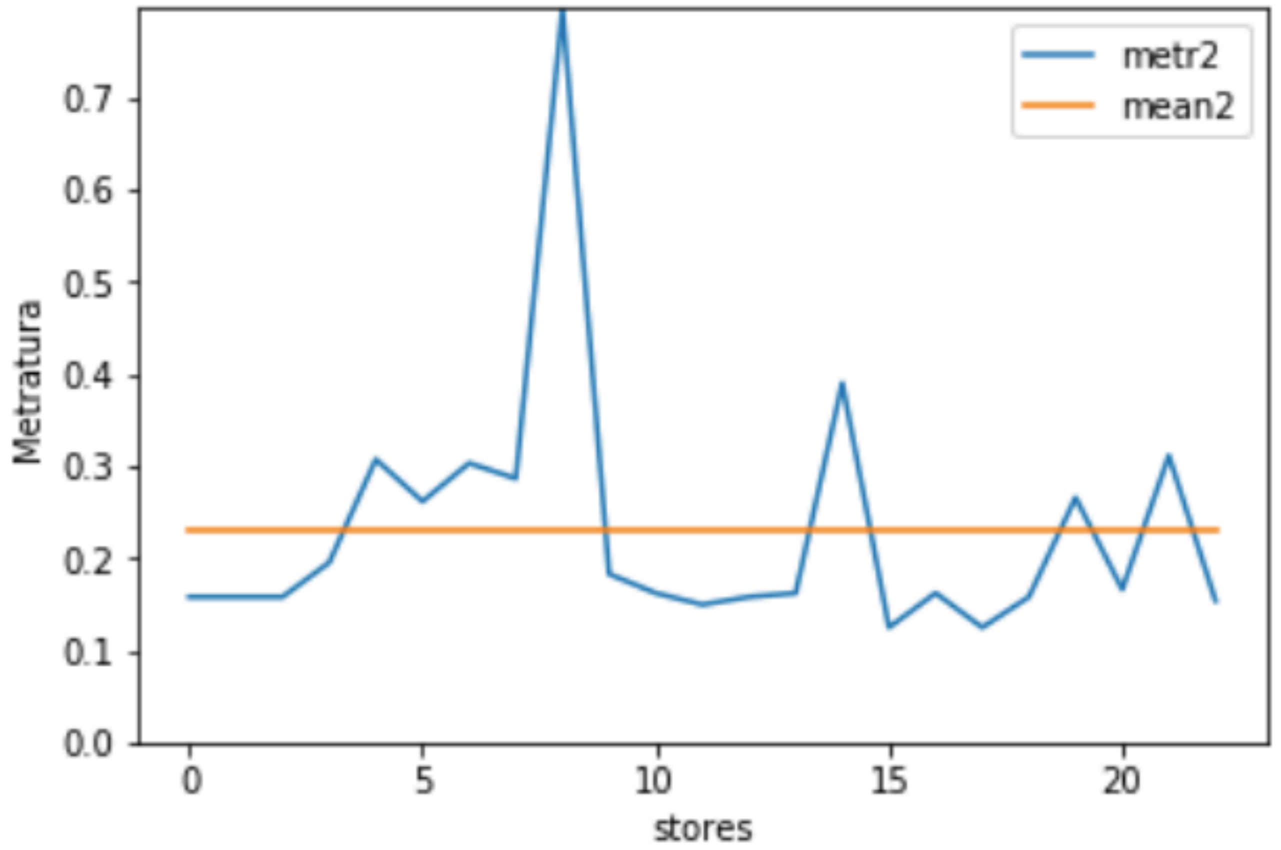
**Compare stores with high footfall against low footfall…to give business solutions!**

# Metratura 0 vs Metratura 2

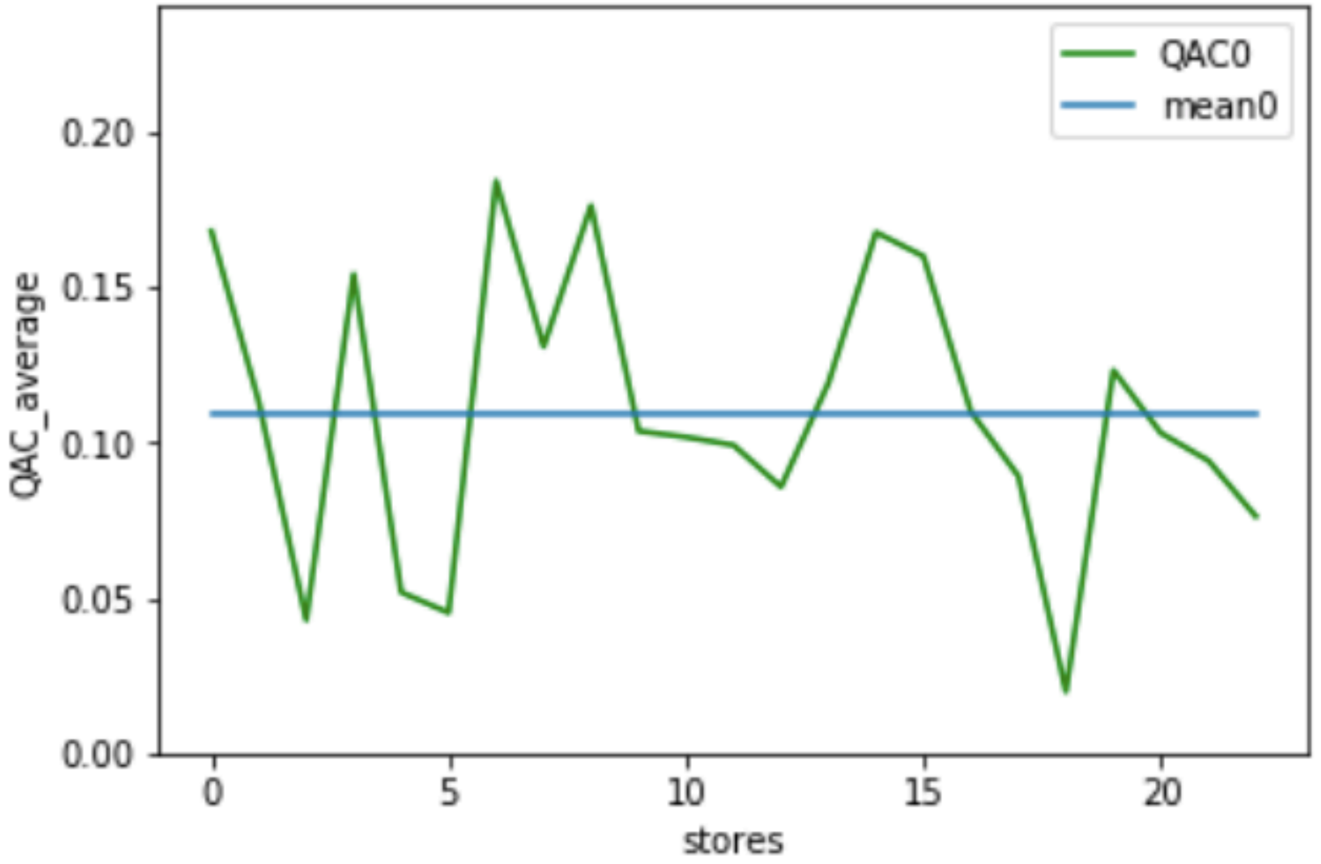



Mean_Metratura_0
0.213335

Mean_Metratura_2
0.230019

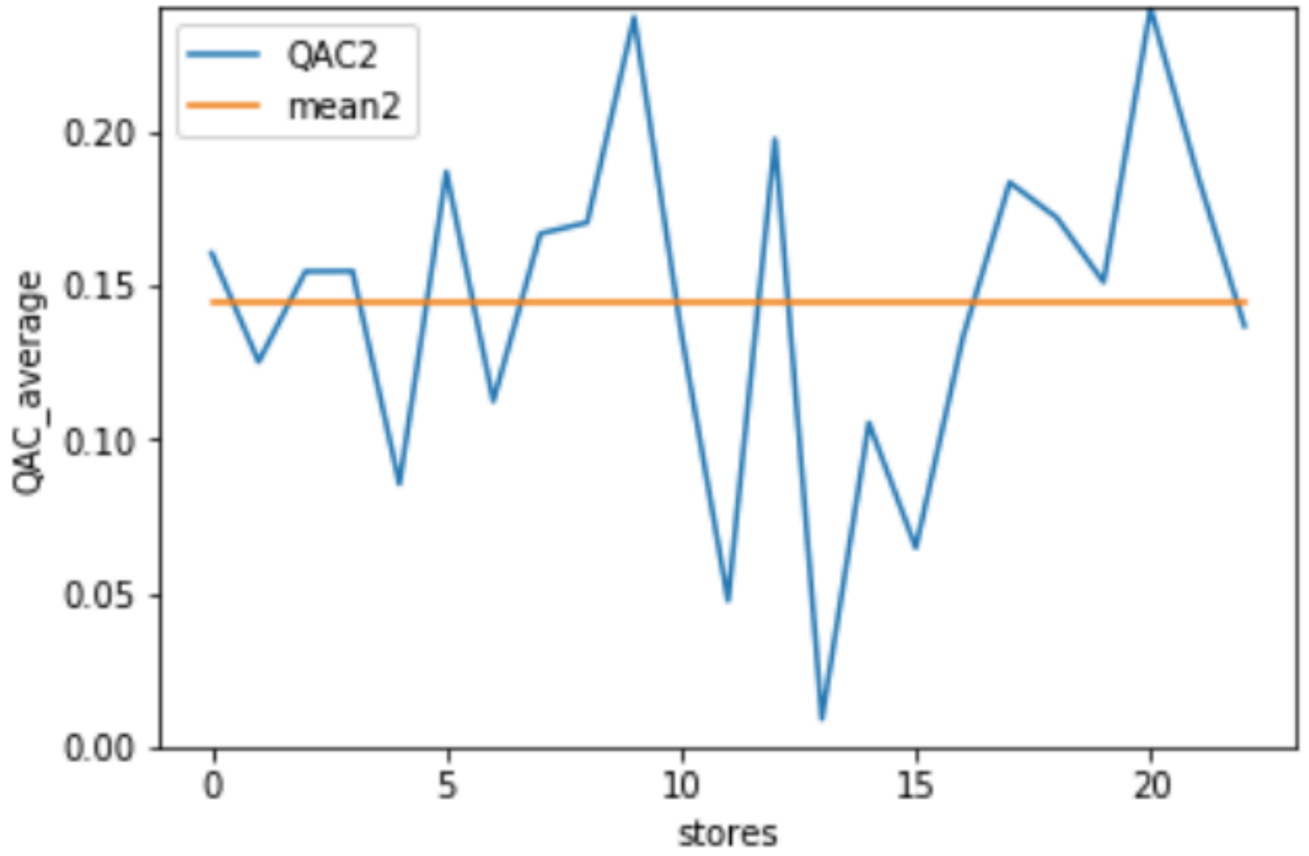


# QAC 0 vs QAC 2

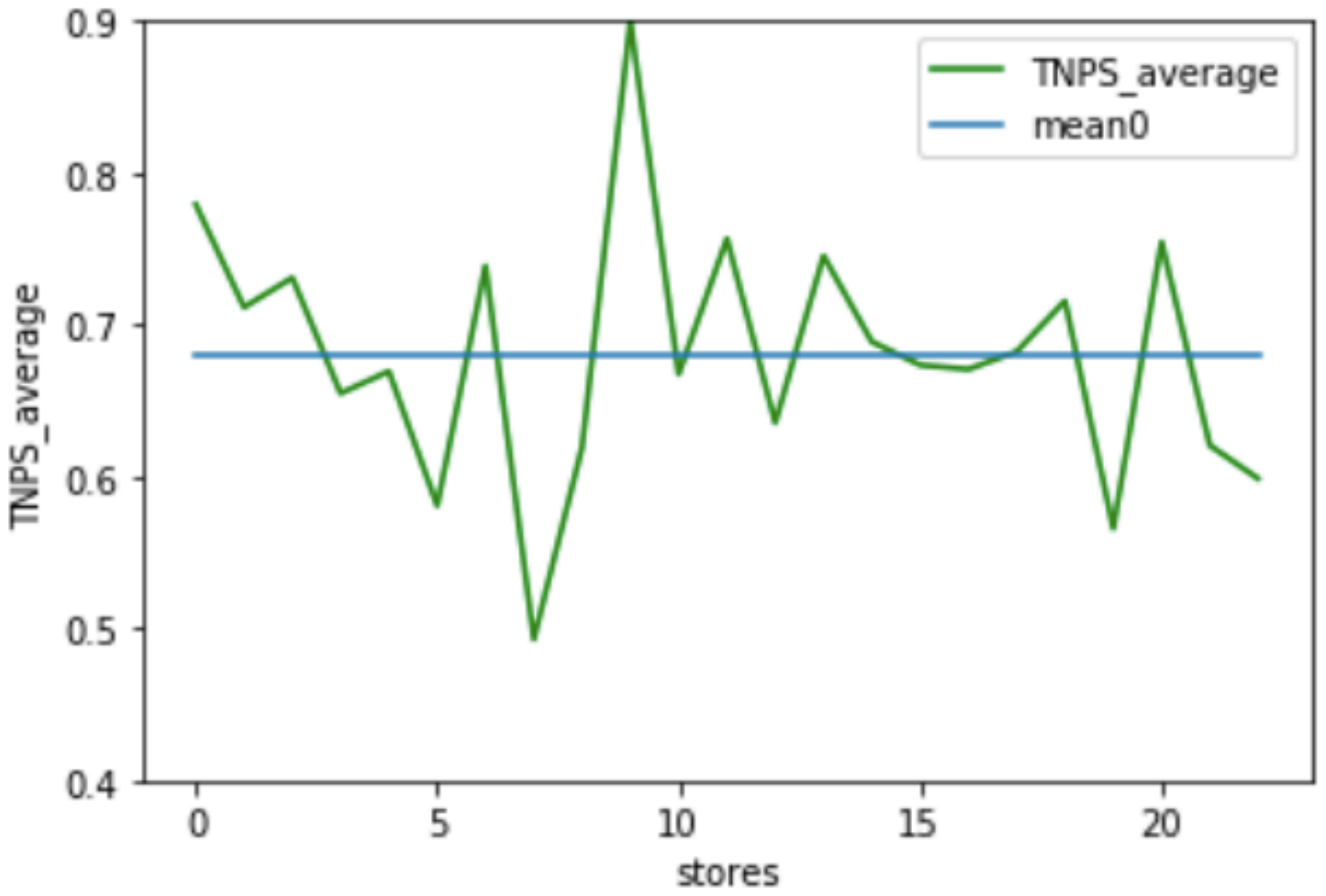



Mean_QAC_0
0.1098

Mean_QAC_2
0.14407

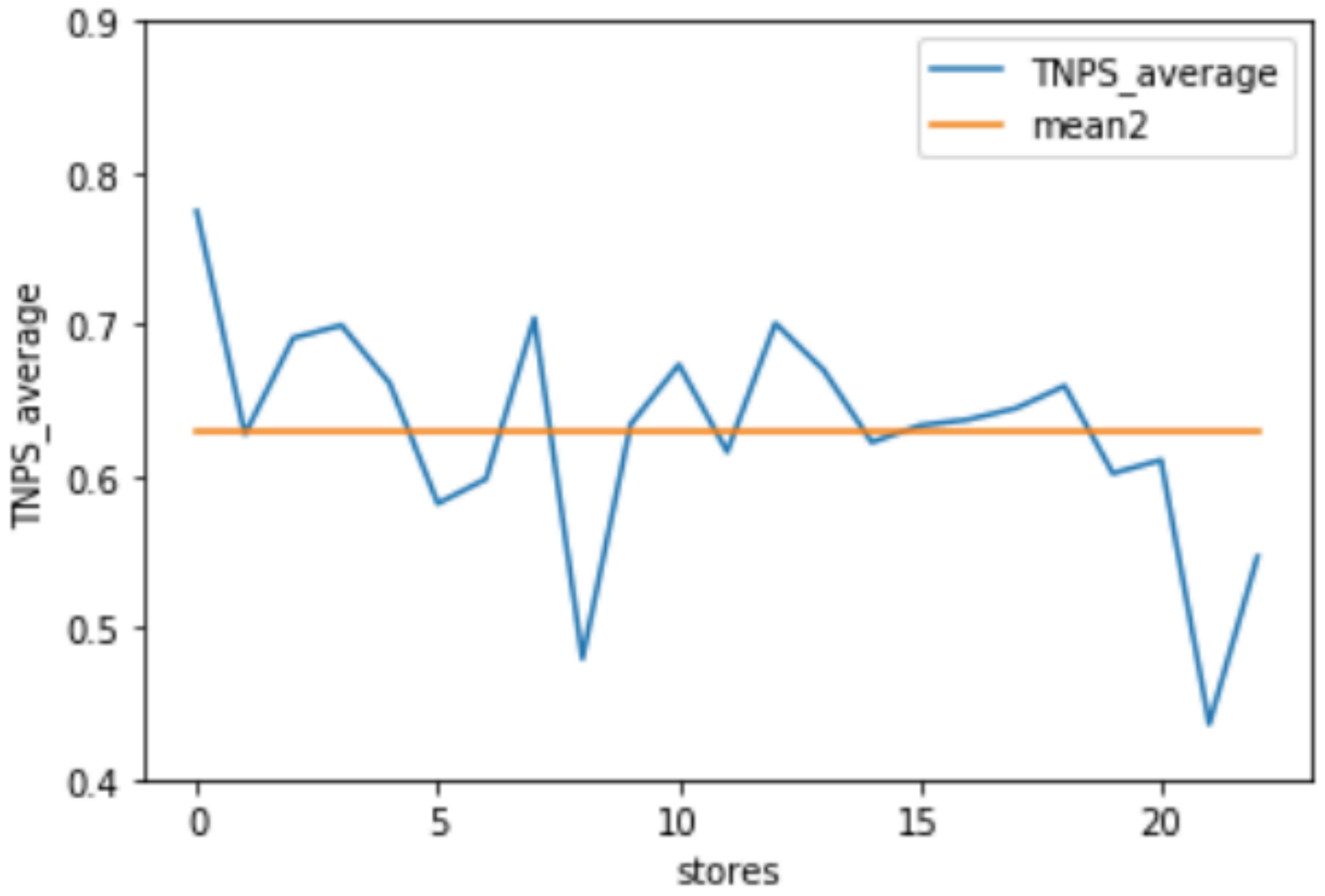


# TNPS 0 vs TNPS 2

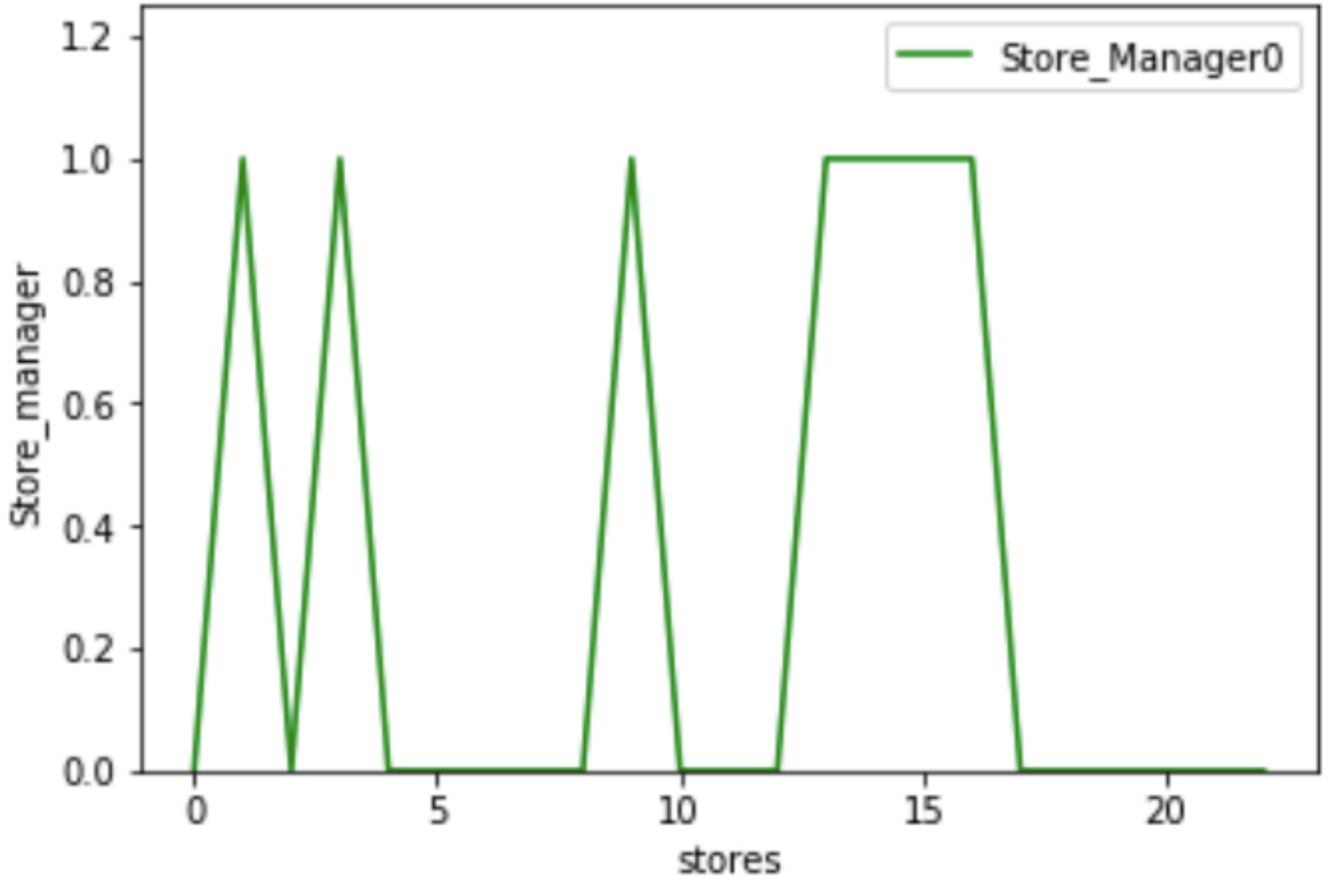



Mean_TNPS_0
0.692260

Mean_TNPS_2
0.6305

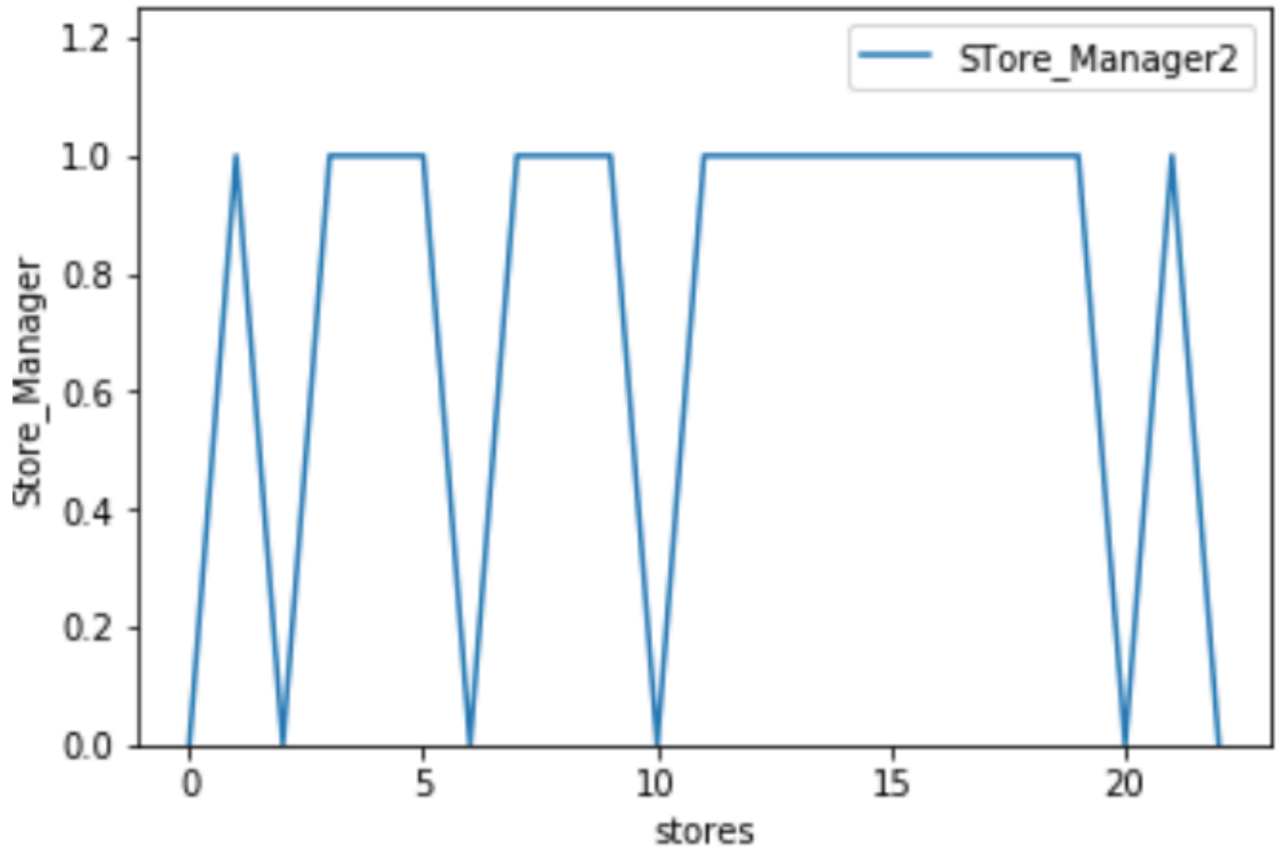


# Manager 0 vs Manager 2

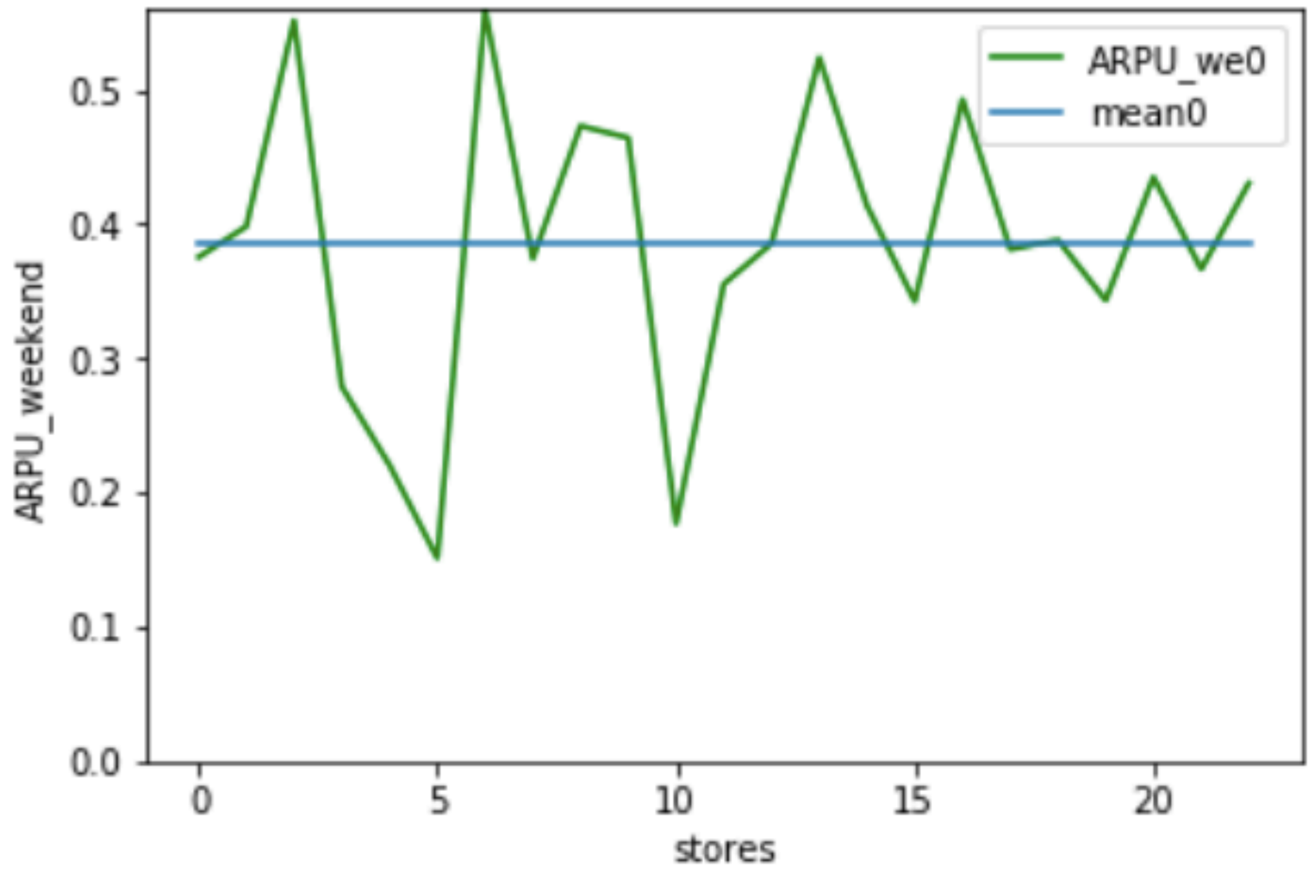



Sum_Manager_0
8

Sum_Manager_2
17

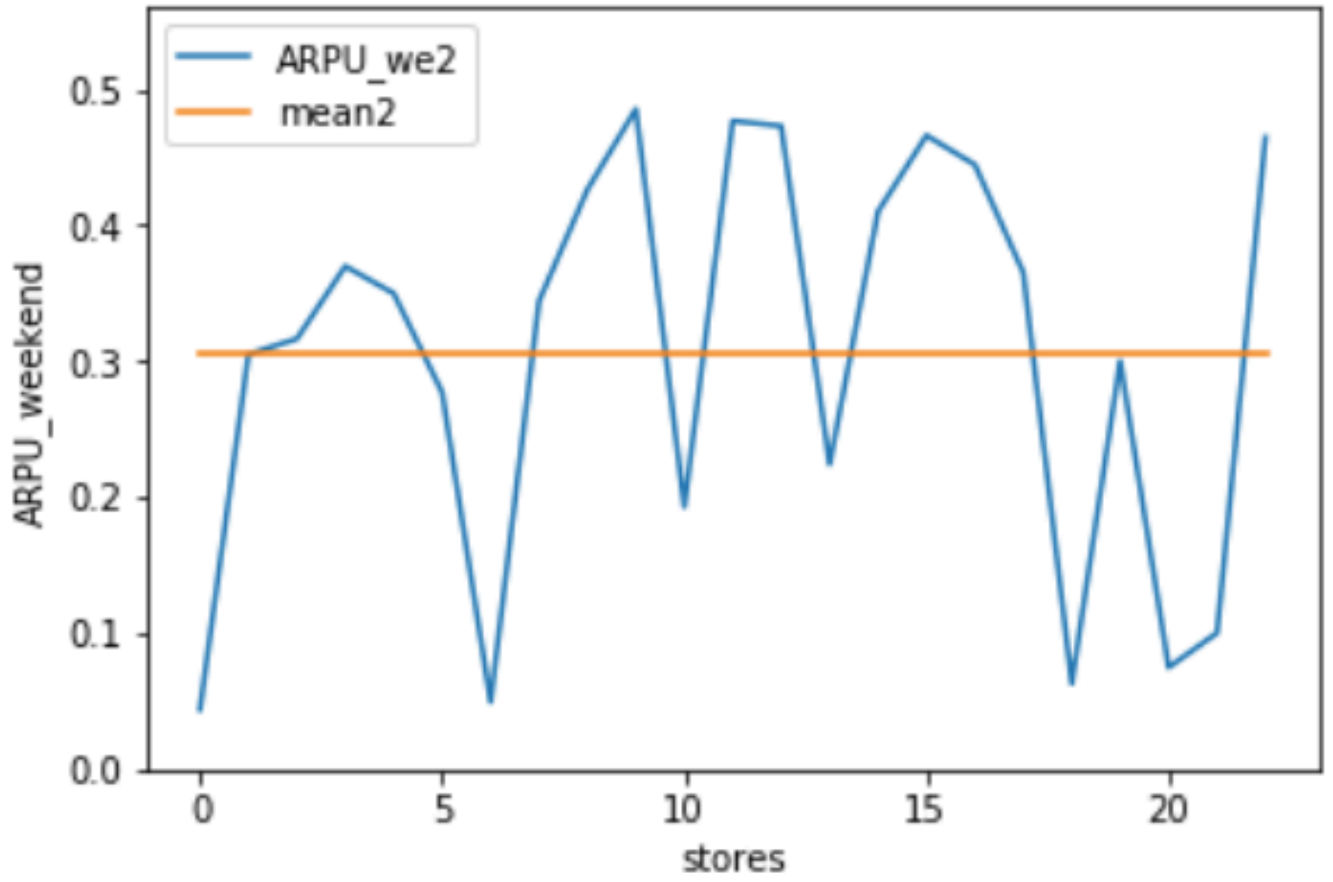


# ARPU_WE 0 vs ARPU_WE 2




Mean_Weekend_ARPU
0.305503

Mean_Weekend_ARPU
0.386400



# Suggested Business Solutions

1. Don’t increase Vodafone Store square meters
2. Increase Number of tickets opened for Phone Assistance
3. Don’t put too much attention on the grade/opinion on stores of VF customer
4. Promote more Store Managers
5. Work on increasing Weekend ARPU!

**Thank you!**