Variation of a company's LinkedIn likes with days of the week.

Ultimately, the goal of brands on LinkedIn is to be able to engage with their customers or potential customers and to promote a message. Thus, it becomes imperative knowing when potential custumers are likeley to be interested in taking a look at their LinkedIn pages. Knowing when users are engaging and interacting with company's page can be crucial to getting the most effective message across.

The goal of this project is to determine how a company's LinkedIn page likes varies from one day of the week to the other and if a mathematical function could be used to approximate such variation.

The analysis is for top ten companies selected from Fortune 500, in addition to a few other very popular social media companies.

Result indicates that polynomial of second oder (quadratic) describes the relationship

Data is from the link: https://thedataincubator.us8.list-manage.com/subscribe/confirm?u=70e04e2160786cdebf3df2567&id=fbf1336bda&e=b835ffc04e (https://thedataincubator.us8.list-manage.com/subscribe/confirm?u=70e04e2160786cdebf3df2567&id=fbf1336bda&e=b835ffc04e)

Imports modules needed

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import calendar
import time
from scipy.optimize import curve_fit
```

Reads the csv data and adds new columns containing the week and month of each row

In [2]:
```
df = pd.read_csv("temp_datalab_records_linkedin_company.csv",low_memory=Fals
df["as_of_date"] = pd.to_datetime(df["as_of_date"],format="%Y-%m-%d")
df['day_of_week'] = df['as_of_date'].apply(lambda x: x.weekday()) # get the
df['day_of_week'] = df['day_of_week'].apply(lambda x: calendar.day_name[x])
df['month'] = df.as_of_date.dt.month

```

Get change in likes where two successive dates are consecutive.

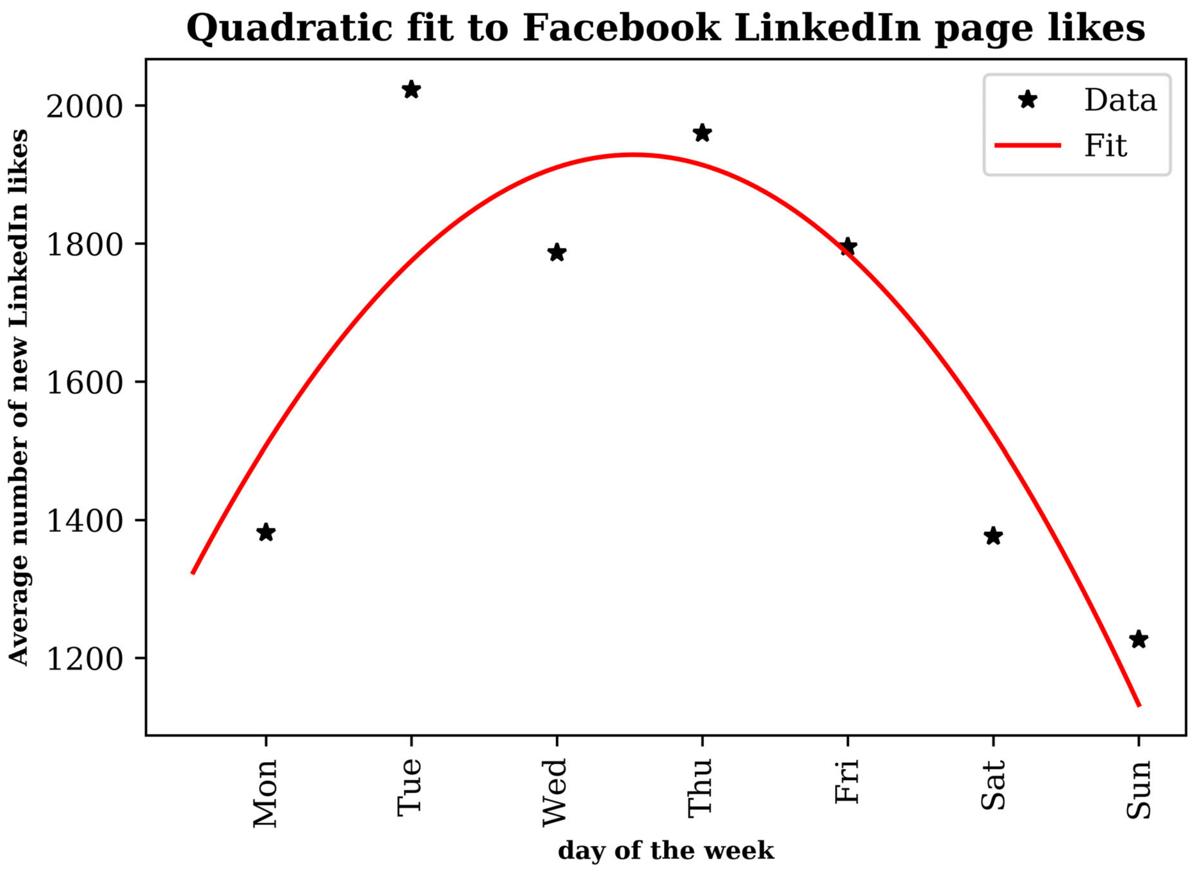
In [3]:
```

def get_change_in_likes(df):
    #converts dates to ordinal for easy computation
    df['ordinal_date'] = df['as_of_date'].apply(lambda x: x.toordinal())
    df["day_difference"] = np.nan
    df["like_difference"] = np.nan
    df["employees_on_platform_difference"] = np.nan
    row_iterator = df.iterrows()
    _, row = next(row_iterator)  # take first item from row_iterator
    for i, _next in row_iterator:
        current_row = row['ordinal_date']
        current_likes = row['followers_count']
        current_employ_likes = row['employees_on_platform']

        next_row = _next['ordinal_date']
        next_likes = _next['followers_count']
        next_employ_likes = _next['employees_on_platform']
        current_and_next_low_list = [current_row,next_row]
        row = _next
        #Checks if two neighboring dates are consecutive
        if max(current_and_next_low_list) - min(current_and_next_low_list) =
        len(current_and_next_low_list) - 1:
            df.loc[i, 'day_difference'] = next_row - current_row
            df.loc[i, "like_difference"] = next_likes - current_likes
            df.loc[i, "employees_on_platform_difference"] = \
            abs(next_employ_likes - current_employ_likes)

        else:
            pass
    # selects rows where day_difference is not null. They satisfy what we wa
    df = df[(df["day_difference"].notnull())]
    df = df[(df["like_difference"].notnull())]
    df = df[(df["employees_on_platform_difference"].notnull())]
    return(df)
```

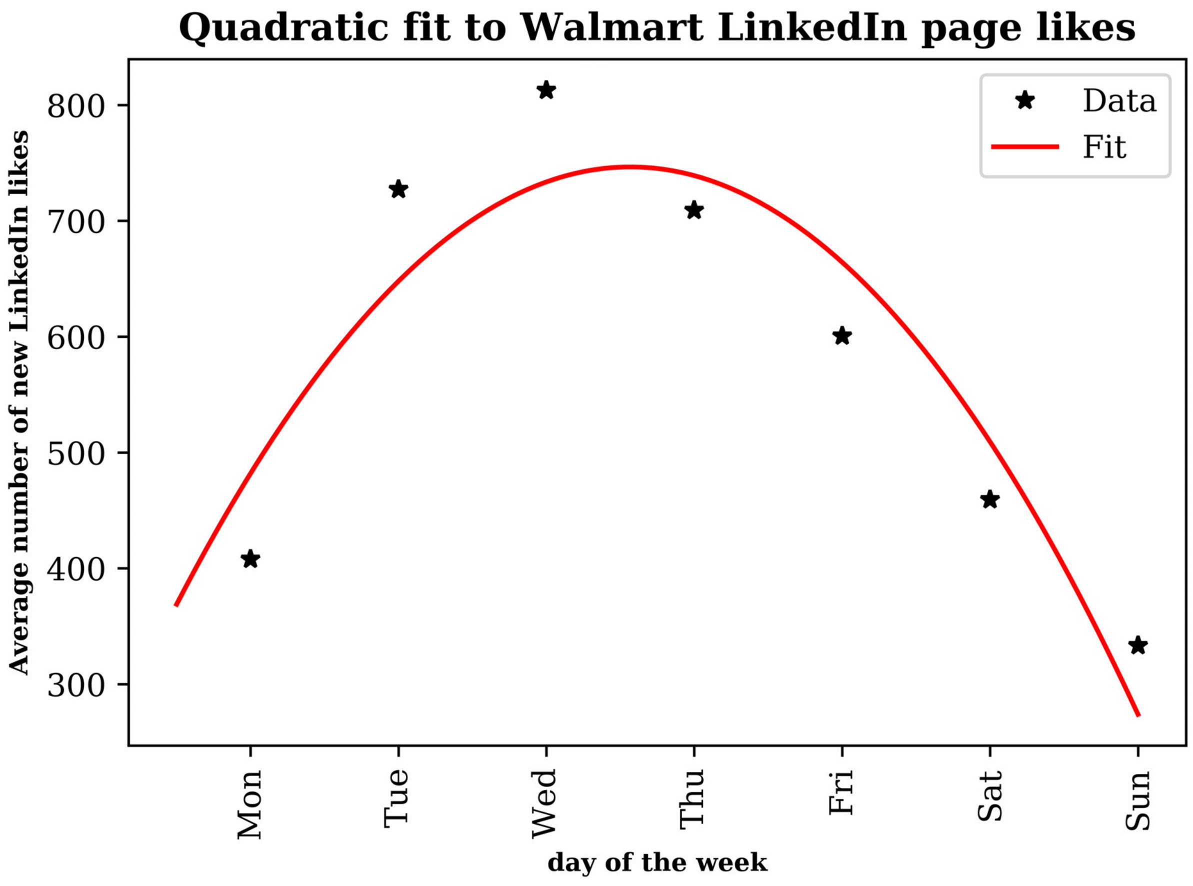
Fits and plots the data and quadratic fit

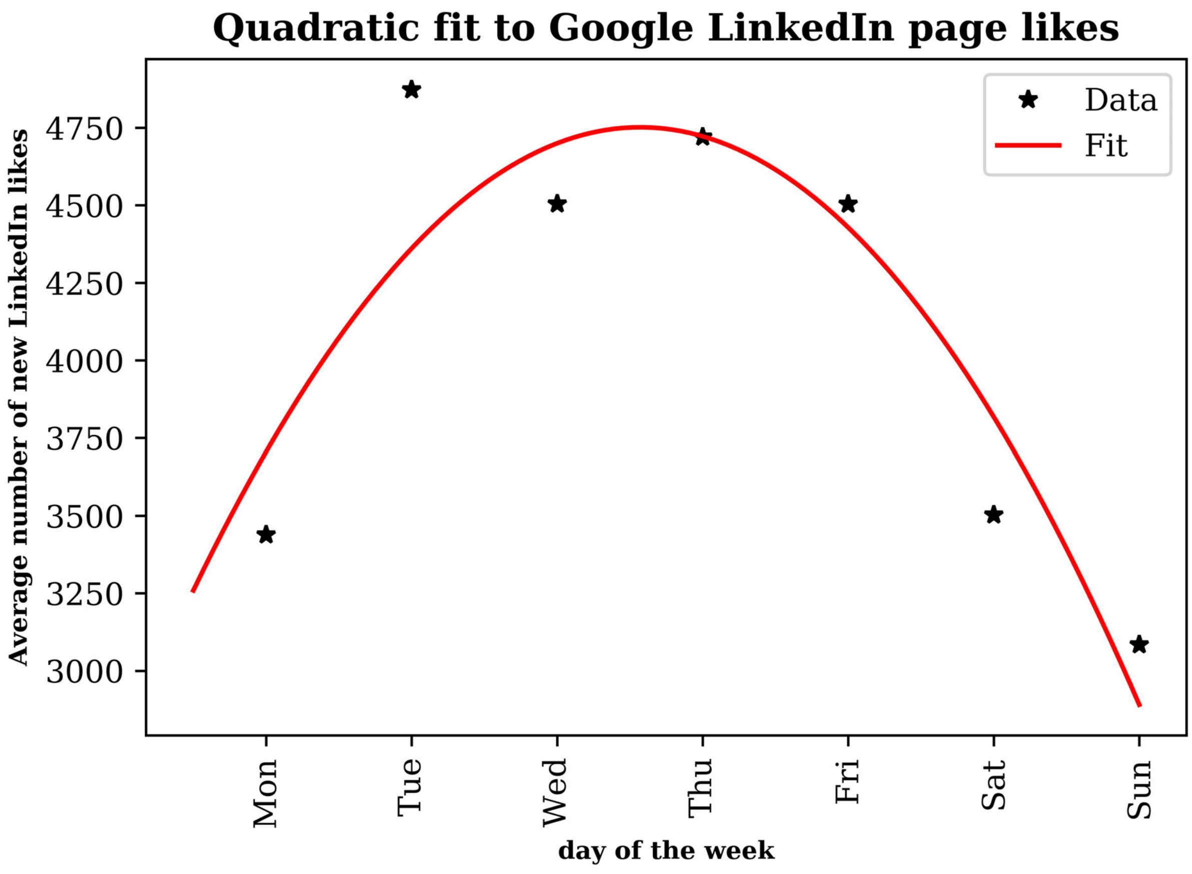
In [ ]:

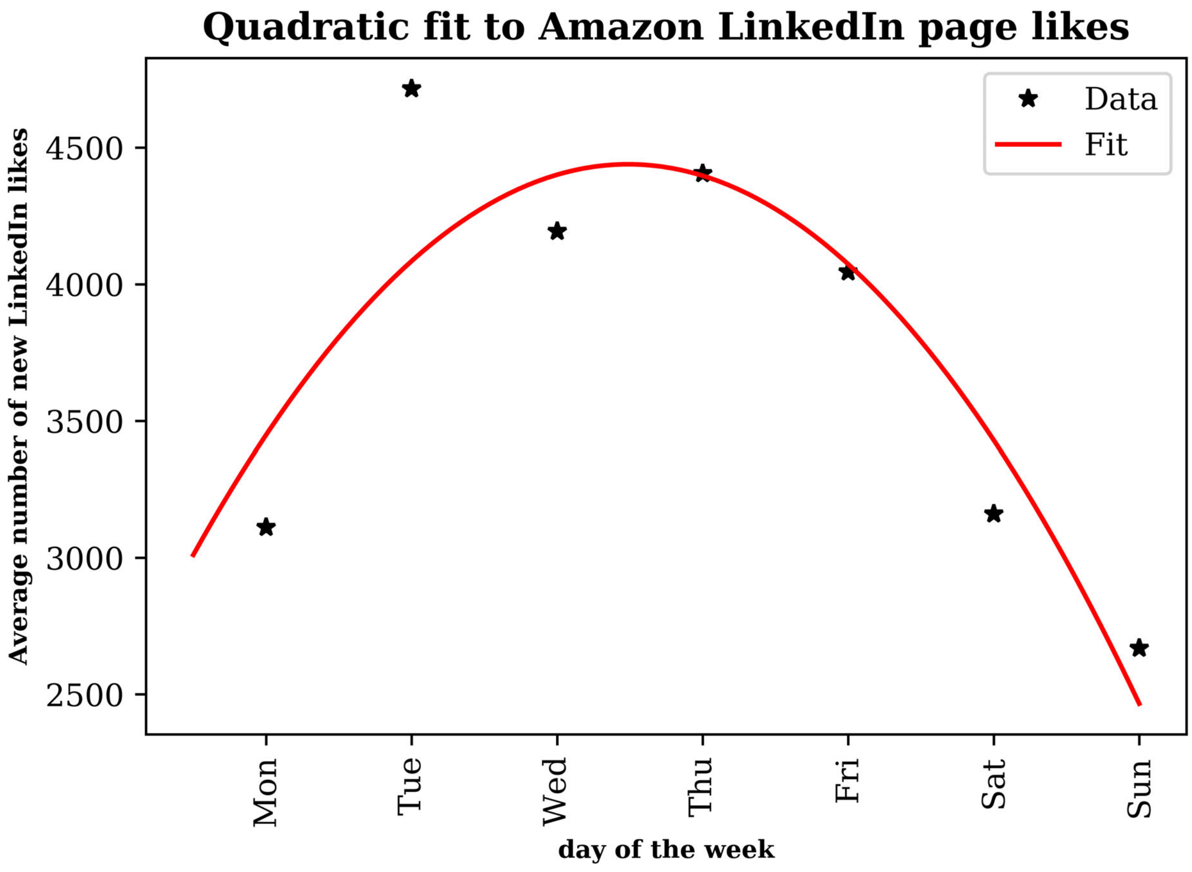
```
plt.rc('font', family='serif')

def quadratic_fit(x,a,b,c):
    return np.array(a+b*x+c*x**2)
groups = df.groupby("company_name")
#company = "Walmart" # put the name of company you want
companies = ["Facebook","Walmart","Google","Amazon","Apple","AT&T","CVS Heal
             "General Motors","UnitedHealth Group","McKesson","ExxonMobil","
for company in companies:
    df1 = groups.get_group(company)
    df1 = df1[df1['followers_count']>=1]
    df1 = get_change_in_likes(df1)
    df1 = df1.groupby("day_of_week")
    #get the days and sort them in the right order
    weekdays = df1.groups.keys()
    days = ["Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","S
    days_abbrev = ["Mon","Tue","Wed","Thu","Fri","Sat","Sun"]
    ordered_weekdays = sorted(weekdays, key=days.index)
    followers_weekly_avg = []
    employ_likes_weekly_avg = []
    for day in ordered_weekdays:
        d = df1.get_group(day)
        mean_following = d.like_difference.mean()
        mean_employ_likes =d.employees_on_platform_difference.mean()
        followers_weekly_avg.append(mean_following)
        employ_likes_weekly_avg.append(mean_employ_likes)
    points = [1,2,3,4,5,6,7] #the days of the week
    P0 = np.array([1,1,1])
    coeffs, matcov = curve_fit(quadratic_fit, points, followers_weekly_avg,
    x = np.linspace(0.5,7,100)
    y = quadratic_fit(x,*coeffs)
    #plots the data
    plt.plot(points,followers_weekly_avg,"k*",x,y,"r")
    plt.xticks(points, days_abbrev)
    plt.title("Quadratic fit to "+company+" LinkedIn page likes",size=12,wei
    plt.xlabel("day of the week",size=8,weight ='bold')
    plt.ylabel("Average number of new LinkedIn likes",size=8,weight ='bold')
    plt.legend(["Data","Fit"])
    plt.xticks(rotation=90)
    plt.savefig(company+".png",bbox_inches="tight", dpi=1000)
    plt.show()
```

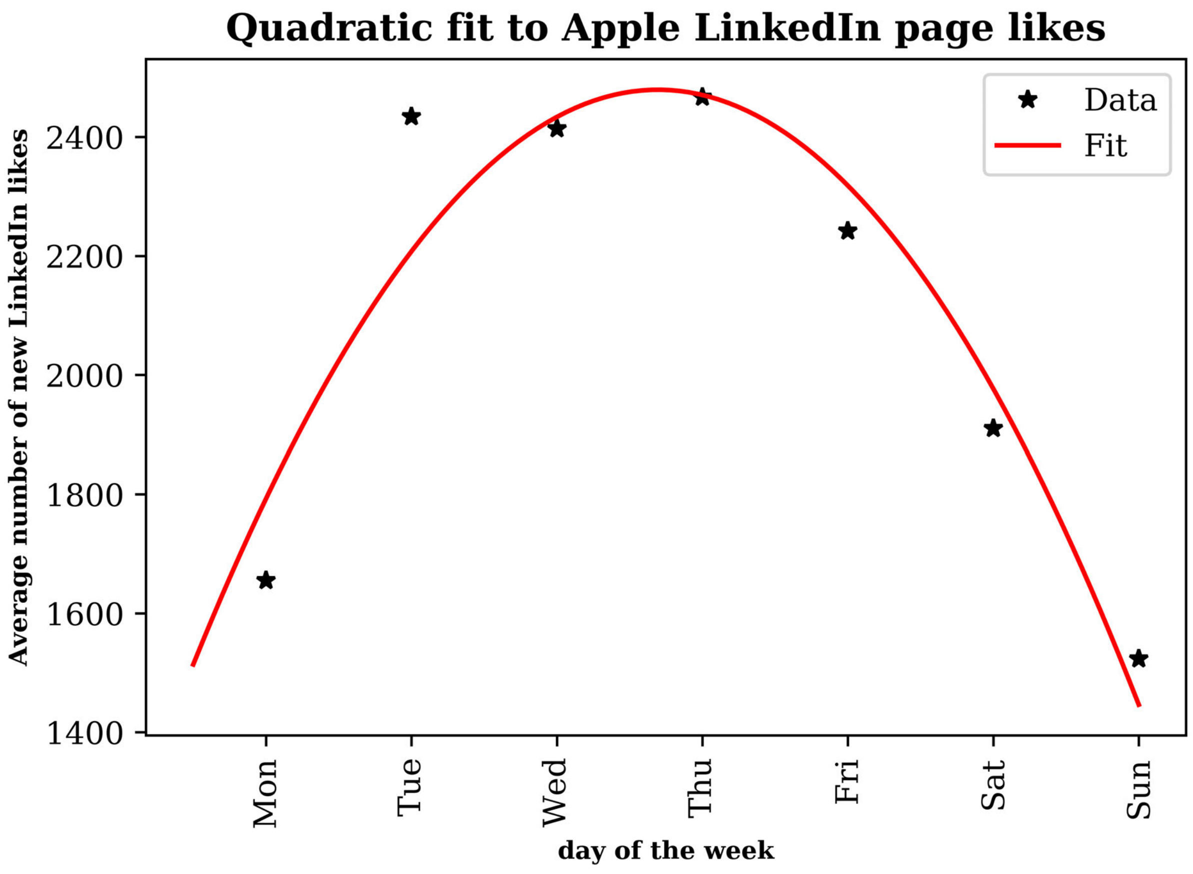
Plots below shows the quadratic relationship between days of the week and average number of new LinkedIn likes

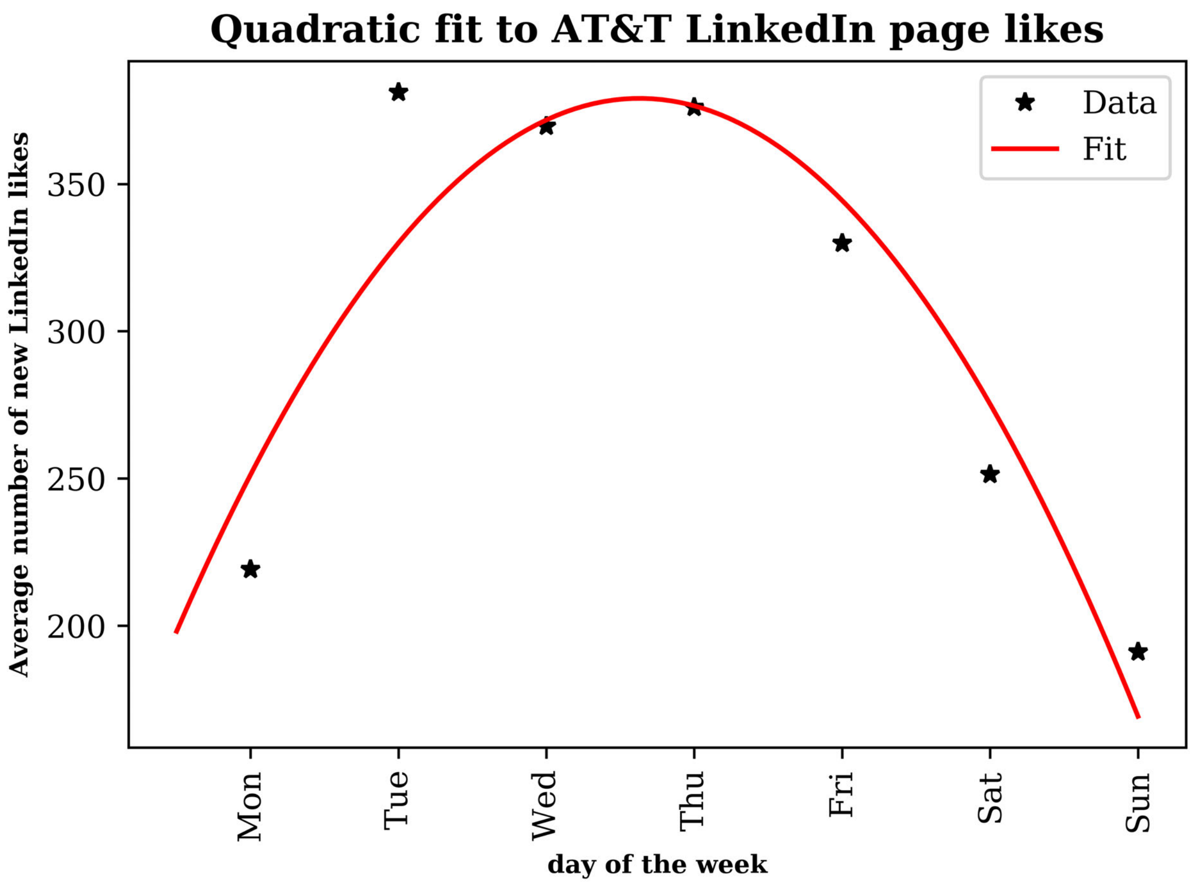

Quadratic fit to Facebook LinkedIn page likes
Average number of new LinkedIn likes
day of the week
Data
Fit
2000
1800
1600
1400
1200
Mon
Tue
Wed
Thu
Fri
Sat
Sun

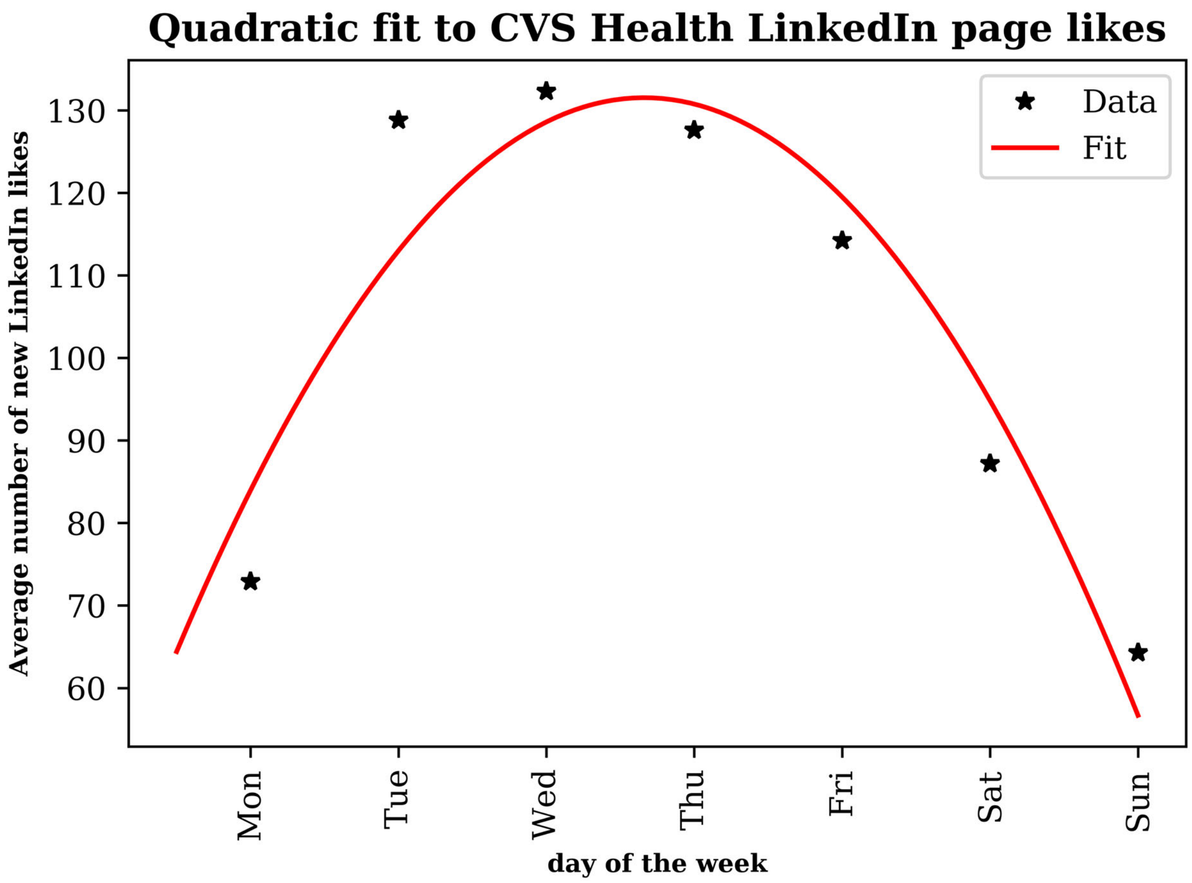

Quadratic fit to Walmart LinkedIn page likes
Average number of new LinkedIn likes
day of the week
Data
Fit
800
700
600
500
400
300
Mon
Tue
Wed
Thu
Fri
Sat
Sun

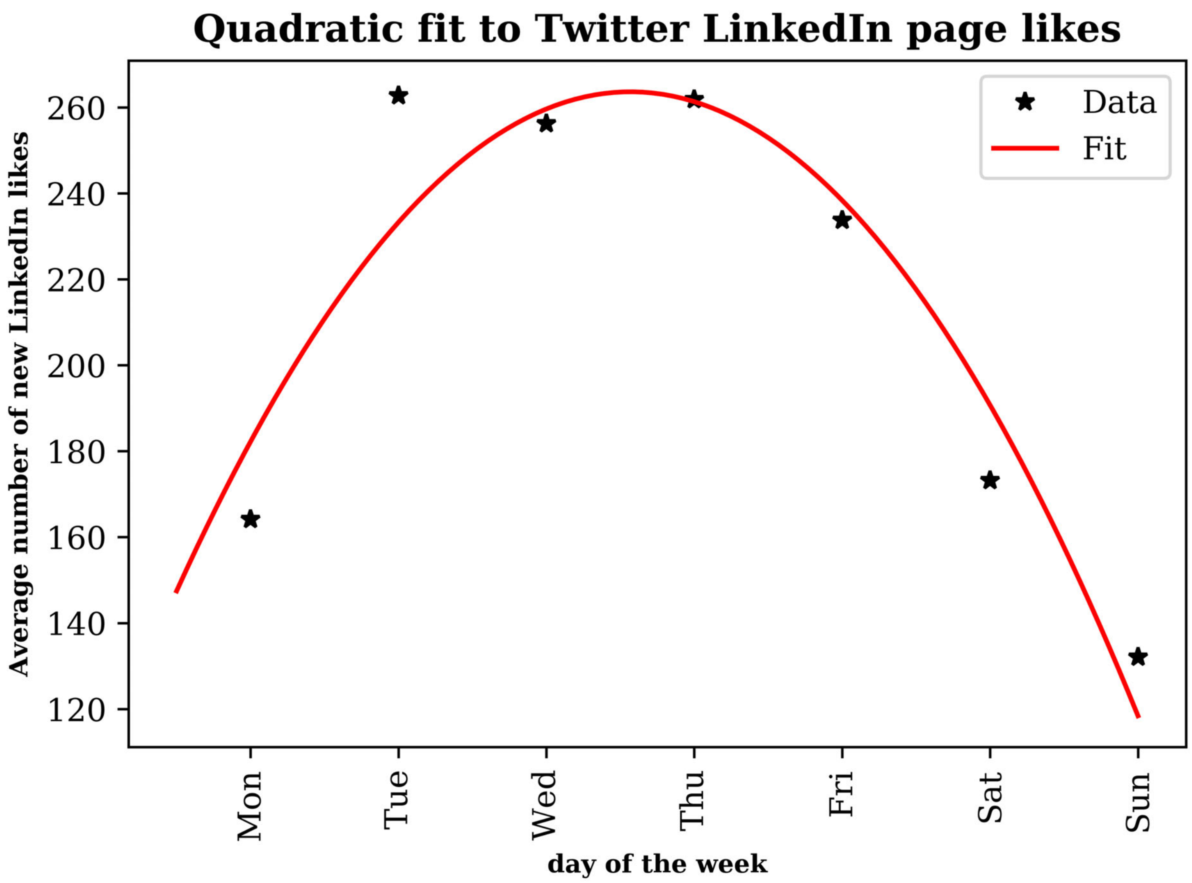

Quadratic fit to Google LinkedIn page likes
Average number of new LinkedIn likes
4750
4500
4250
4000
3750
3500
3250
3000
Mon
Tue
Wed
Thu
Fri
Sat
Sun
day of the week
Data
Fit

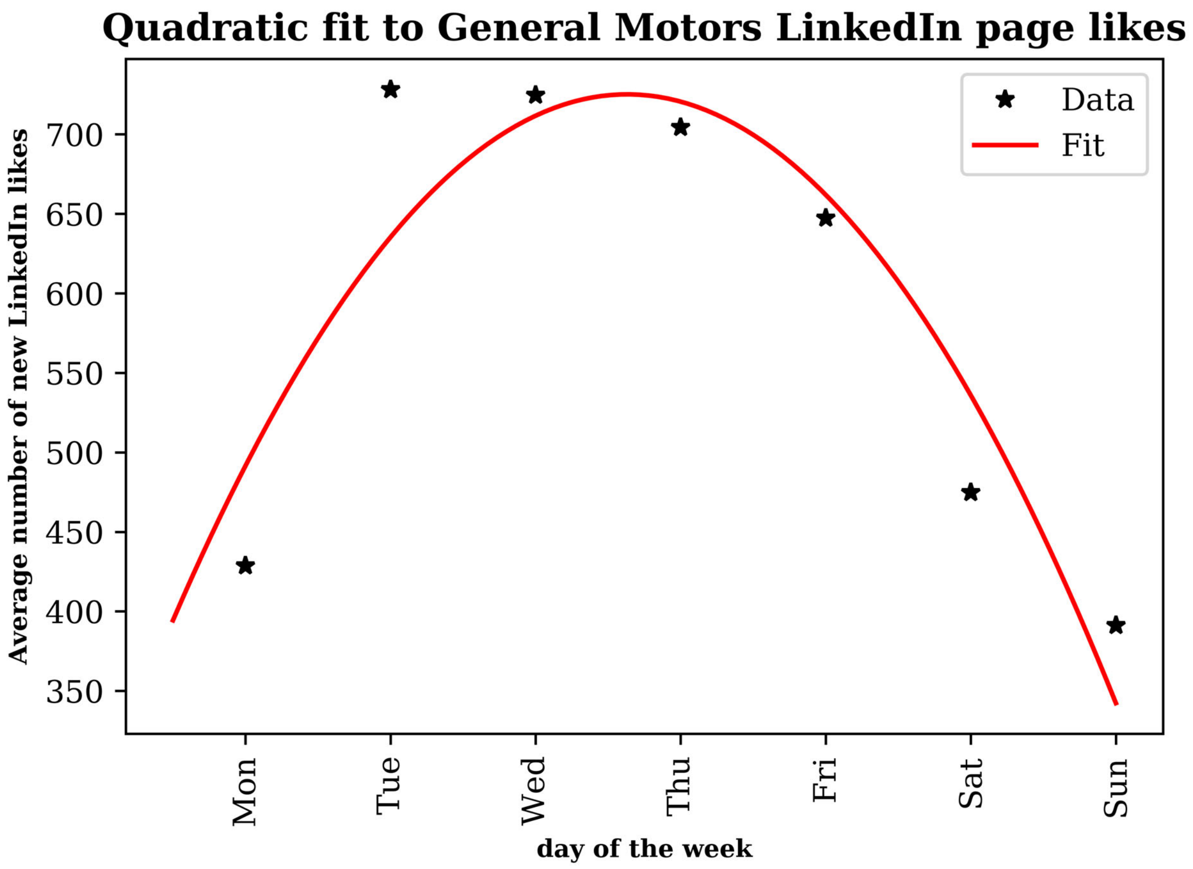

Quadratic fit to Amazon LinkedIn page likes
Average number of new LinkedIn likes
4500
4000
3500
3000
2500
Mon
Tue
Wed
Thu
Fri
Sat
Sun
day of the week
Data
Fit

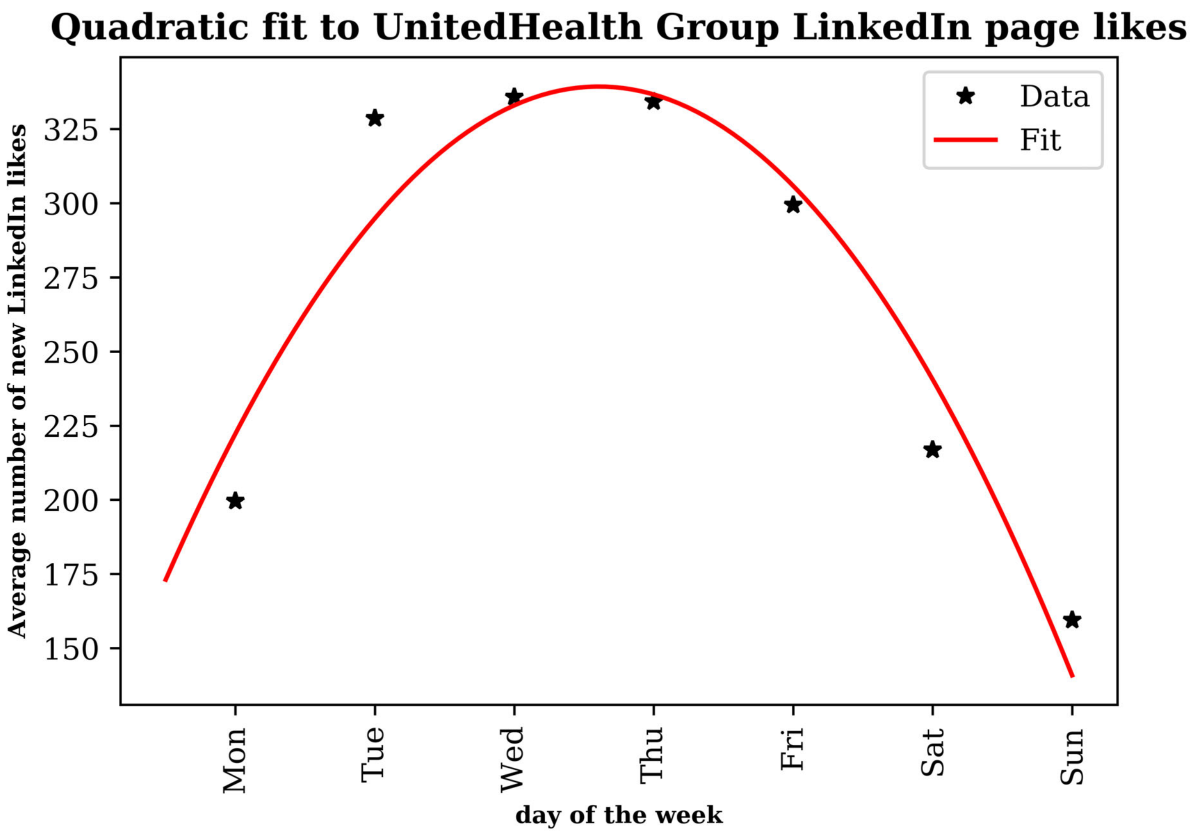

Quadratic fit to Apple LinkedIn page likes
Average number of new LinkedIn likes
1400
1600
1800
2000
2200
2400
Mon
Tue
Wed
Thu
Fri
Sat
Sun
day of the week
Data
Fit

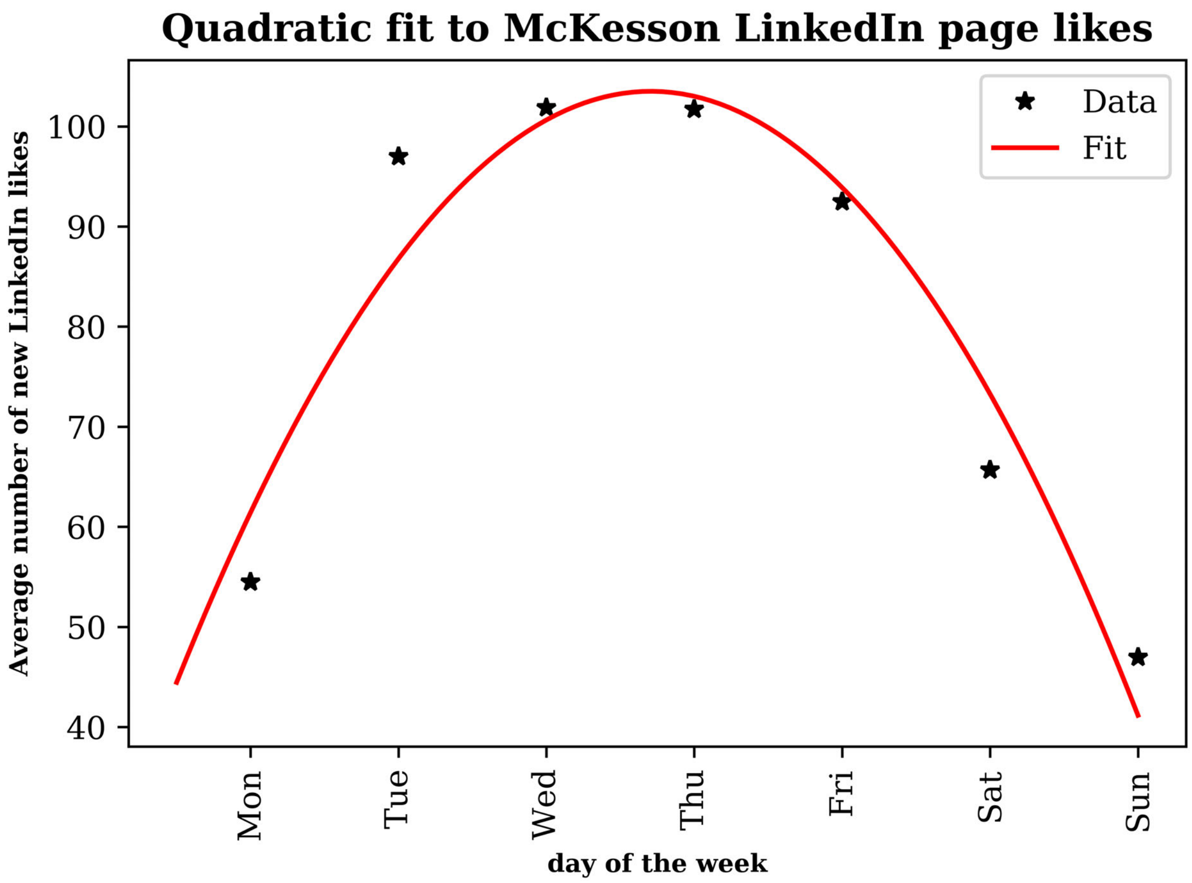

Quadratic fit to AT&T LinkedIn page likes
Average number of new LinkedIn likes
200
250
300
350
Mon
Tue
Wed
Thu
Fri
Sat
Sun
day of the week
Data
Fit

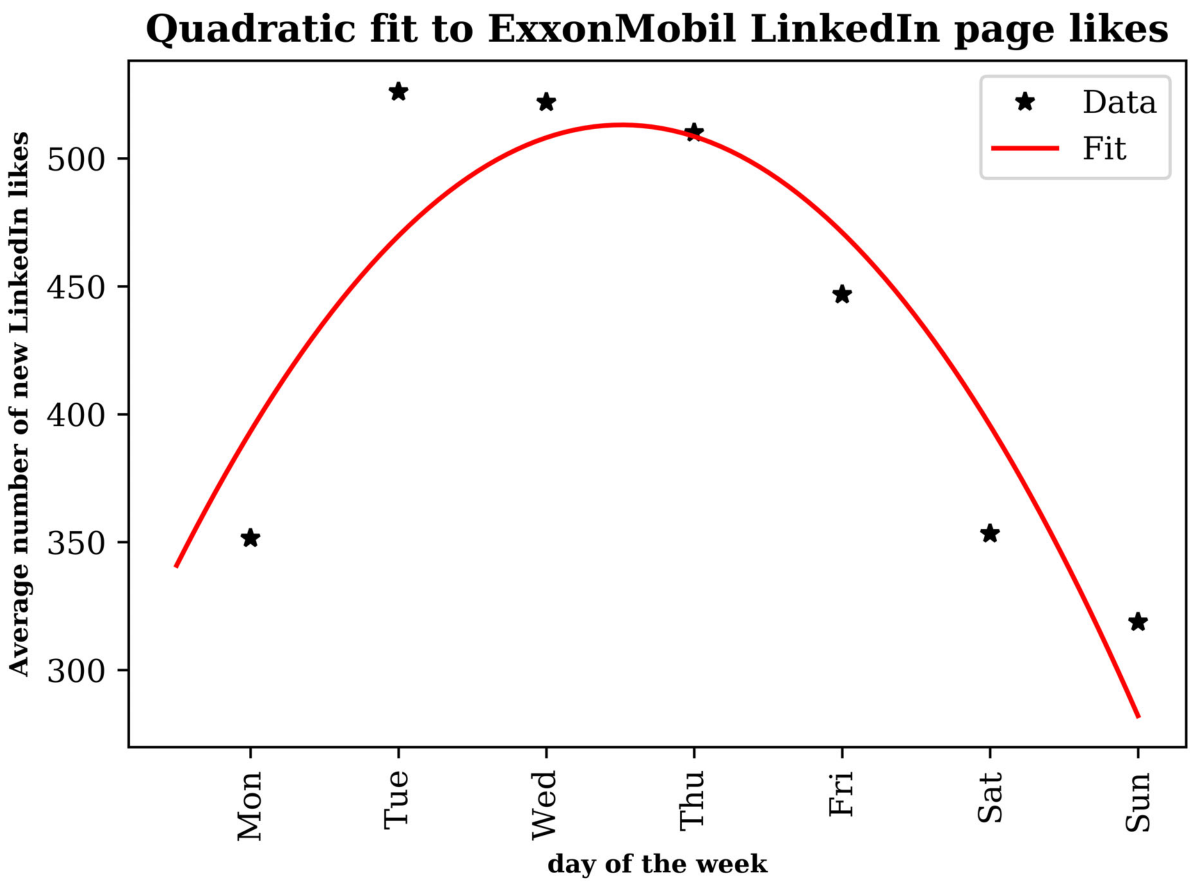

Quadratic fit to CVS Health LinkedIn page likes
Average number of new LinkedIn likes
day of the week
Data
Fit
Mon
Tue
Wed
Thu
Fri
Sat
Sun
60
70
80
90
100
110
120
130

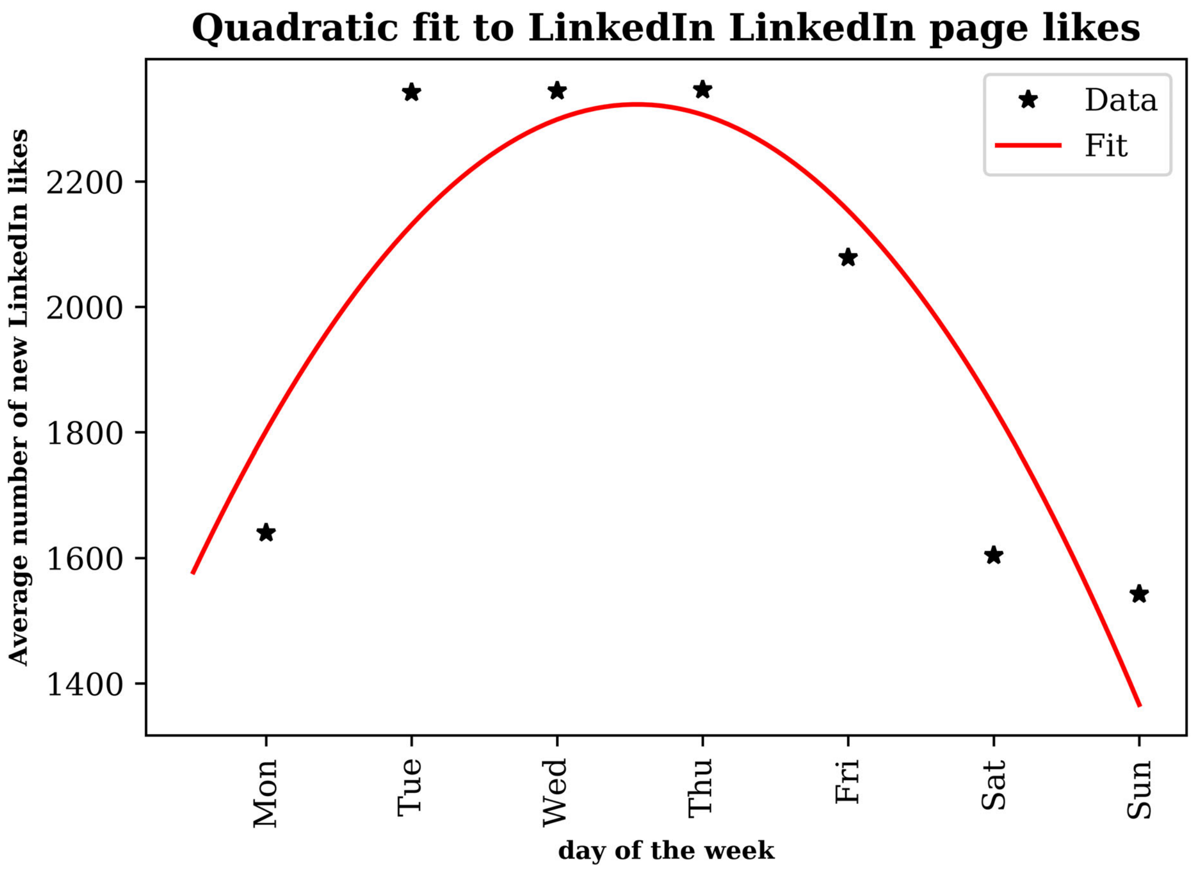

Quadratic fit to Twitter LinkedIn page likes
Average number of new LinkedIn likes
day of the week
Data
Fit
Mon
Tue
Wed
Thu
Fri
Sat
Sun
120
140
160
180
200
220
240
260


Quadratic fit to General Motors LinkedIn page likes
Average number of new LinkedIn likes
day of the week
Data
Fit
Mon
Tue
Wed
Thu
Fri
Sat
Sun
350
400
450
500
550
600
650
700


Quadratic fit to UnitedHealth Group LinkedIn page likes
Average number of new LinkedIn likes
day of the week
Data
Fit
Mon
Tue
Wed
Thu
Fri
Sat
Sun
150
175
200
225
250
275
300
325


Quadratic fit to McKesson LinkedIn page likes
Data
Fit
Average number of new LinkedIn likes
40
50
60
70
80
90
100
Mon
Tue
Wed
Thu
Fri
Sat
Sun
day of the week


Quadratic fit to ExxonMobil LinkedIn page likes
Data
Fit
Average number of new LinkedIn likes
300
350
400
450
500
Mon
Tue
Wed
Thu
Fri
Sat
Sun
day of the week



As can be see, increase in likes peaks at midweek. So, those companies will better engage with potential customers if they advertise on LinkedIn in the midweek.

This reference: [https://mashable.com/2010/10/28/facebook-activity-study/#RX35mrR835q8 (https://mashable.com/2010/10/28/facebook-activity-study/#RX35mrR835q8)](https://mashable.com/2010/10/28/facebook-activity-study/#RX35mrR835q8), done using facebook data supports the result from this analysis.