In [1]:
```
import numpy as np
import scipy as sc
from scipy.integrate import odeint
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score

import calendar
from datetime import datetime, date

import pandas as pd
```

read_file = pd.read_excel ('DBM PKL Astro (1).xlsx', sheet_name='Sheet2') read_file.to_csv ('inputcme.csv', index = None, header=True)

In [2]:
```
#INITIAL INPUT PARAMETERS
Rs= 20                                  #in unit of solar radius
R0 = Rs*695700                             # unit is Km
V0= 319                                   # unit is km/s

R_target= 1.0                              # target distance in AU
w= 645                                      # which is ambient solar wind speed in unit of km/s
Gamma=0.2
gamma=Gamma*10**(-7)                       # unit is km-1
Time_UTC=datetime(2012,10,27,16,54,0)    #input utc time in format (year,month,date,hr,minute,second)
```

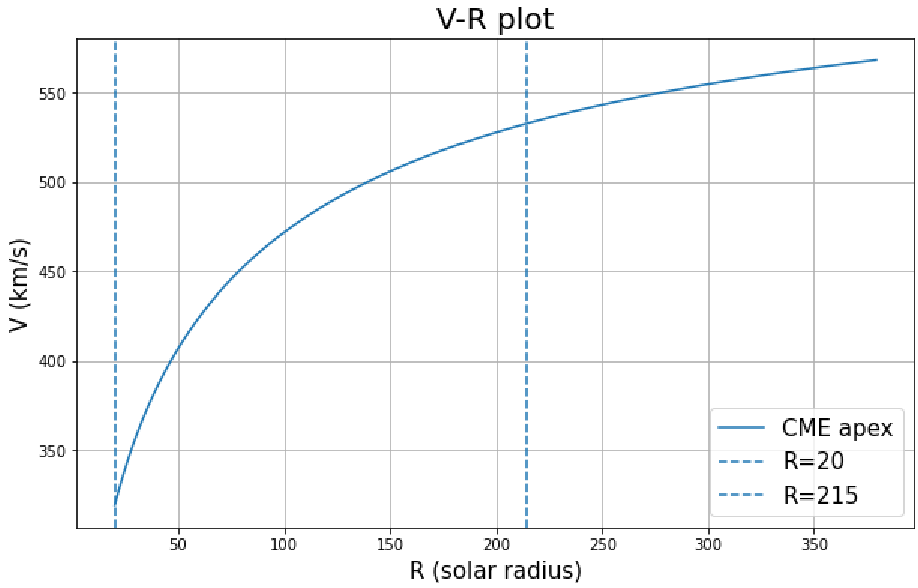
In [3]:
```
# defining function for equation of motion

def dbm(x,t):
    r,v=x
    dxdt=[v,-gamma*(v-w)*np.abs(v-w)]
    return dxdt
```

In [4]:
```
# Solving equation of motion with given boundary conditions.
import datetime
ts = calendar.timegm(Time_UTC.timetuple()) #this command provide second correspond to given input time
t=np.arange(0,500000,0.1)
Time=[Time_UTC + datetime.timedelta(milliseconds=i*10) for i in range(len(t))]
Y0=[R0,V0]


Y=odeint(dbm,Y0,t)

R=Y[:,0]/695700             # from now onwards we take solar radius as unit of distance
V=Y[:,1]
```

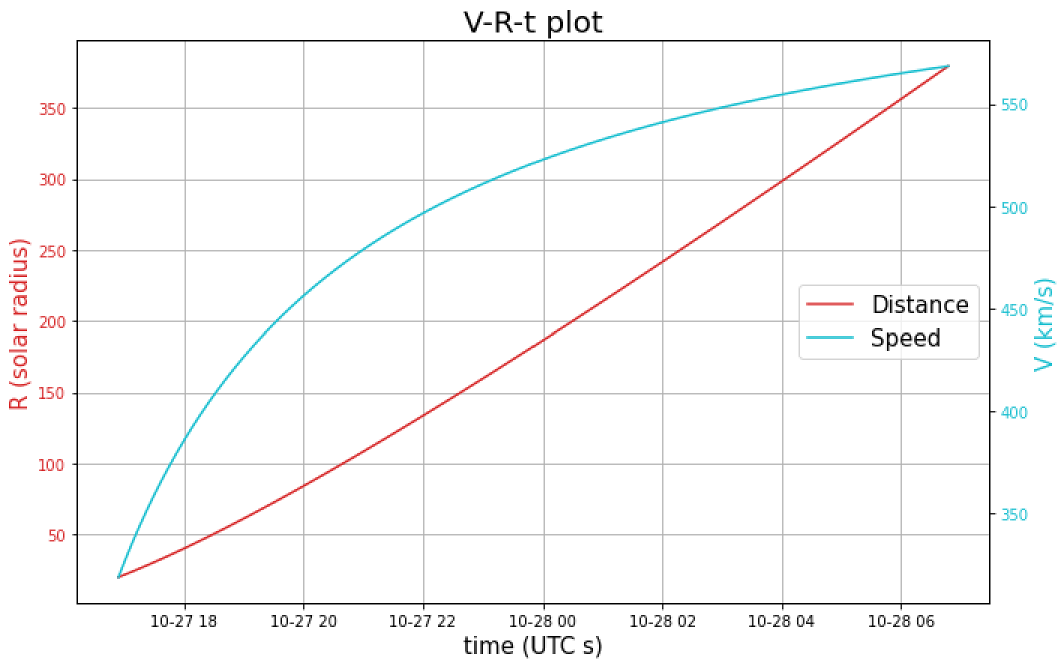
In [5]:
```
#FORECASTING
def find_nearest(d,v, value):
    from datetime import datetime, date
    array = np.asarray(d)
    idx = (np.abs(array - value)).argmin()
    v=float("{:.3f}".format(v[idx]))
    T=float("{:.3f}".format((t[idx]-t[0])/3600))
    T_Utc_date=datetime.utcfromtimestamp(t[idx]).strftime('%d %b %Y')
    T_Utc_time = datetime.utcfromtimestamp(t[idx]).time().strftime('%H:%M:%S')
    return T,v,T_Utc_date,T_Utc_time

A=find_nearest(R/215,V,R_target)

print("Transit time of CME is " +str(A[0]) + " hr")
print("Imapact speed of CME at "+str(R_target)+ " AU is " +str(A[1]) +" Km/s")
print("CME arrives at "+str(R_target)+" AU (date & time) "+ str(A[2])+ " at " + str(A[3]) )
```

```
Transit time of CME is 81.436 hr
Imapact speed of CME at 1.0 AU is 533.029 Km/s
CME arrives at 1.0 AU (date & time) 04 Jan 1970 at 09:26:09
```

In [6]:
```
#PLOT
plt.figure(figsize=(10,6))
plt.plot(R,V,label="CME apex")
plt.axvline(x=20,linestyle="dashed",label="R=20")
plt.axvline(x=214,linestyle="dashed",label="R=215")
plt.xlabel("R (solar radius)",fontsize=15)
plt.ylabel("V (km/s)",fontsize=15)
plt.legend(fontsize=15)
plt.title("V-R plot", fontsize=20)
plt.grid()
```



In [7]:
```
fig, ax1 = plt.subplots(figsize=(10,6))
plt.grid()
color = 'tab:red'
ax1.set_xlabel('time (UTC s)', fontsize=15)
ax1.set_ylabel('R (solar radius)', color=color,fontsize=15)

ax1.plot(Time, R, color=color,label="Distance")
ax1.tick_params(axis='y', labelcolor=color)

ax2 = ax1.twinx()  # instantiate a second axes that shares the same x-axis

color = 'tab:cyan'
ax2.set_ylabel('V (km/s)', color=color,fontsize=15)  # we already handled the x-label with ax1
ax2.plot(Time, V, color=color,label="Speed")
ax2.tick_params(axis='y', labelcolor=color)

fig.tight_layout()  # otherwise the right y-label is slightly clipped

lines_1, labels_1 = ax1.get_legend_handles_labels()
lines_2, labels_2 = ax2.get_legend_handles_labels()

lines = lines_1 + lines_2
labels = labels_1 + labels_2

ax1.legend(lines, labels,fontsize=15,loc=5)

plt.title("V-R-t plot", fontsize=20)
plt.show()
```



In [8]:
```
#Gamma variasi
```

```
DBM = pd.read_csv("inputcme.csv")
DBM
```

Out[8]:

	v0	gamma	w	tt	v1	tt_calc	v1_calc	tt_web	v1_web	tt_ml	v1_ml	Unnamed: 11	Unnamed: 12
0	319	0.14	645	84.83	507	85.395	509.340	85.35	509	76.74	560	NaN	NaN
1	896	0.83	517	87.55	552	59.631	565.885	59.59	566	83.97	564	NaN	NaN
2	220	0.15	786	75.50	616	76.301	616.135	76.26	616	70.38	639	NaN	NaN
3	131	0.41	430	115.00	381	116.201	381.209	116.14	381	101.78	429	NaN	NaN
4	162	0.55	451	104.25	410	105.397	409.896	105.34	410	86.13	496	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
144	1377	0.47	652	78.60	720	41.432	771.196	41.40	771	64.99	786	NaN	NaN
145	710	0.32	616	68.40	605	54.626	675.062	54.59	675	71.81	594	NaN	NaN
146	167	0.19	768	75.62	620	76.049	622.347	76.01	622	67.53	659	NaN	NaN
147	576	0.69	429	101.40	460	75.427	468.156	75.38	468	99.48	456	NaN	NaN
148	0	1.30	577	76.00	550	76.708	550.427	0.00	0	77.64	546	NaN	NaN

149 rows × 13 columns

In [9]:

```
# deleting unnecessary columns
for col in DBM.columns:
    if 'Unnamed' in col:
        del DBM[col]
```

In [10]:

```
DBM
```

Out[10]:

	v0	gamma	w	tt	v1	tt_calc	v1_calc	tt_web	v1_web	tt_ml	v1_ml
0	319	0.14	645	84.83	507	85.395	509.340	85.35	509	76.74	560
1	896	0.83	517	87.55	552	59.631	565.885	59.59	566	83.97	564
2	220	0.15	786	75.50	616	76.301	616.135	76.26	616	70.38	639
3	131	0.41	430	115.00	381	116.201	381.209	116.14	381	101.78	429
4	162	0.55	451	104.25	410	105.397	409.896	105.34	410	86.13	496
...	...	...	...	...	...	...	...	...	...	...	...
144	1377	0.47	652	78.60	720	41.432	771.196	41.40	771	64.99	786
145	710	0.32	616	68.40	605	54.626	675.062	54.59	675	71.81	594
146	167	0.19	768	75.62	620	76.049	622.347	76.01	622	67.53	659
147	576	0.69	429	101.40	460	75.427	468.156	75.38	468	99.48	456
148	0	1.30	577	76.00	550	76.708	550.427	0.00	0	77.64	546

149 rows × 11 columns

In [11]:

```
R0 = 20*695700                #R0 at 20R_sun, in km
tt_calc = []
v1_calc = []
R_target= 1.0 # target distance in AU
for i in range (len(DBM)):
    V0=DBM['v0'][i]
    w= DBM["w"][i]            # which is ambient solar wind speed in unit of km/s
    Gamma=DBM['gamma'][i]              # fix gamma
    gamma=Gamma*10**(-7)
    t=np.arange(0,500000,0.1)
    Y0=[R0,V0]
    Y=odeint(dbm,Y0,t)
    R=Y[:,0]/695700           # from now onwards we take solar radius as unit of distance
    V=Y[:,1]
    A=find_nearest(R/215,V,R_target)
    tt_calc.append(A[0])
    v1_calc.append(A[1])


DBM['tt_calc'] = tt_calc
DBM['v1_calc'] = v1_calc
```

DBM.to_csv('outputcme.csv') read_file = pd.read_csv ("outputcme.csv") read_file.to_excel ("outputcme.xlsx", index = None, header=True)

In [12]:

```
# DIFFERENCE BETWEEN TT_CALCULATED BY PYTHON VS TT_DATA
dttcd = np.abs((DBM['tt_calc']-DBM['tt'])/DBM['tt'])
```

```
# DIFFERENCE BETWEEN V1_CALCULATED BY PYTHON VS V1_DATA
dv1cd = np.abs((DBM['v1_calc']-DBM['v1'])/DBM['v1'])
# DIFFERENCE BETWEEN TT_ML VS TT_DATA
dttmd = np.abs((DBM['tt_ml']-DBM['tt'])/DBM['tt'])
# DIFFERENCE BETWEEN V1_ML VS V1_DATA
dv1md = np.abs((DBM['v1_ml']-DBM['v1'])/DBM['v1'])
# DIFFERENCE BETWEEN TT_ML VS TT_CALC
dttmc = np.abs((DBM['tt_calc']-DBM['tt_ml'])/DBM['tt_ml'])
# DIFFERENCE BETWEEN V1_ML VS V1_CALC
dv1mc = np.abs((DBM['v1_calc']-DBM['v1_ml'])/DBM['v1_ml'])
```

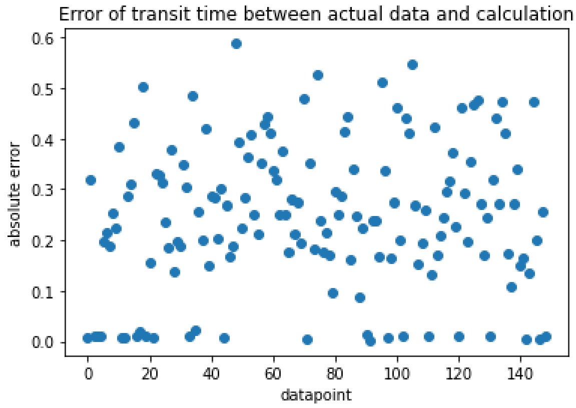
In [13]:

```
# mean error
print("tt actual-calc",np.mean(dttcd))
print("v1 actual-calc",np.mean(dv1cd))
print("tt actual-ml", np.mean(dttmd))
print("v1 actual-ml",np.mean(dv1md))
print("tt ml-calc",np.mean(dttmc))
print("v1 ml-calc",np.mean(dv1mc))
```

```
tt actual-calc 0.24497436507117826
v1 actual-calc 0.03479235785152574
tt actual-ml 0.0898773955228085
v1 actual-ml 0.09235858535132527
tt ml-calc 0.2389601436976708
v1 ml-calc 0.0874461267242171
```

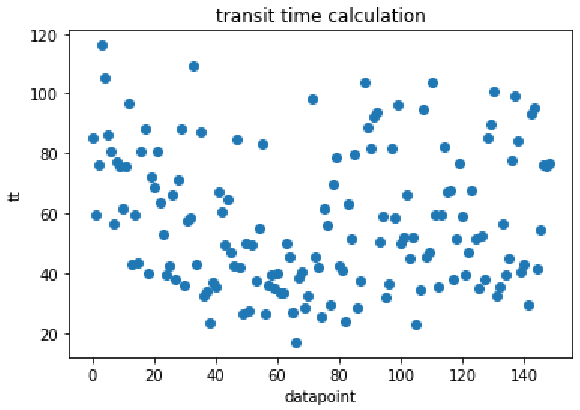
In [14]:

```
plt.scatter(DBM.index,dttcd)
plt.title('Error of transit time between actual data and calculation')
plt.xlabel('datapoint')
plt.ylabel('absolute error')
plt.savefig('transit time error Actual data vs Calc scatter.png')
```



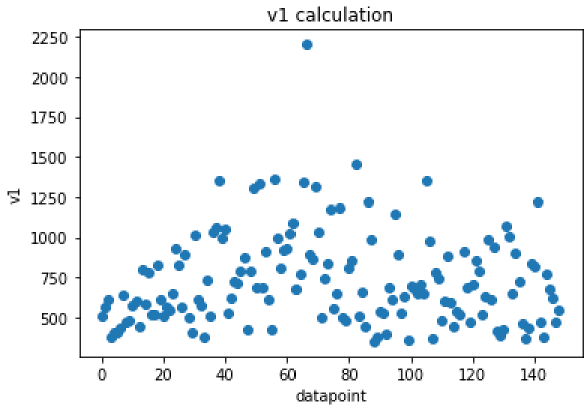
In [15]:

```
plt.scatter(DBM.index,DBM['tt_calc'])
plt.title('transit time calculation')
plt.xlabel('datapoint')
plt.ylabel('tt')
plt.savefig('transit time calc.png')
```

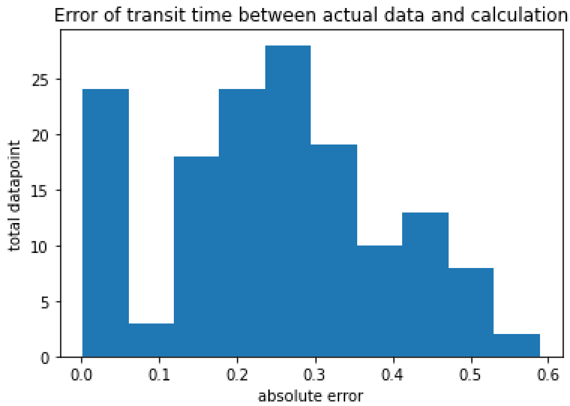


In [16]:

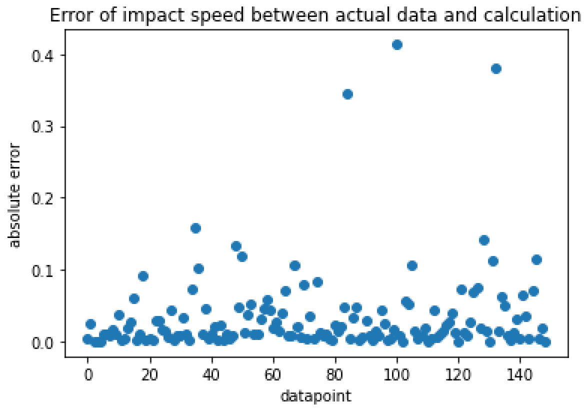
```
plt.scatter(DBM.index,DBM['v1_calc'])
plt.title('v1 calculation')
plt.xlabel('datapoint')
plt.ylabel('v1')
plt.savefig('v1 calc.png')
```



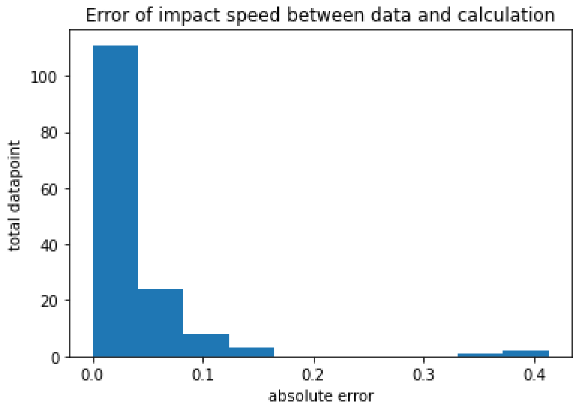
In [17]:
```
plt.hist(dttcd)
plt.title('Error of transit time between actual data and calculation')
plt.ylabel('total datapoint')
plt.xlabel('absolute error')
plt.savefig('transit time error actual data vs calc hist.png')
```



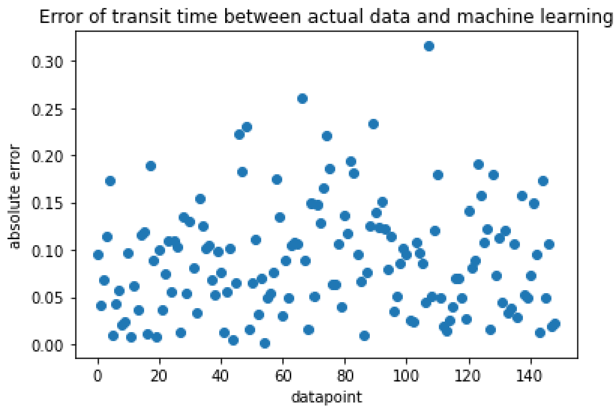
In [18]:
```
plt.scatter(DBM.index,dv1cd)
plt.title('Error of impact speed between actual data and calculation')
plt.xlabel('datapoint')
plt.ylabel('absolute error')
plt.savefig('error of impact speed actual vs calc scatter.png')
```



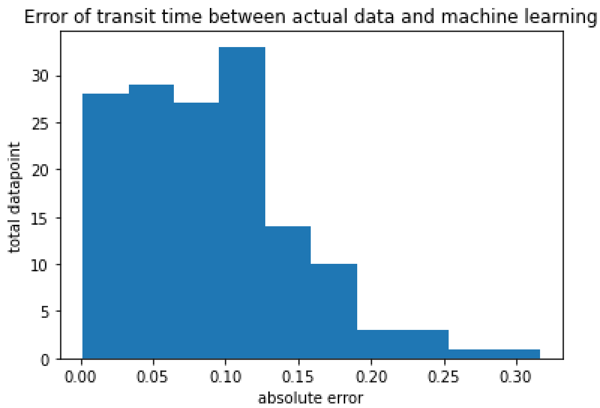
In [19]:
```
plt.hist(dv1cd)
plt.title('Error of impact speed between data and calculation')
plt.ylabel('total datapoint')
plt.xlabel('absolute error')
plt.savefig('error of impact speed data vs calc hist.png')
```



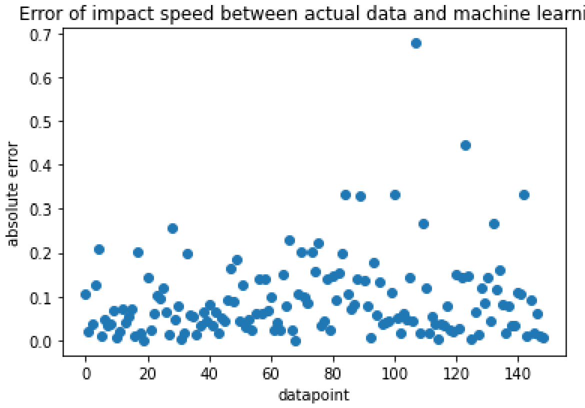
In [20]:
```
plt.scatter(DBM.index,dttmd)
plt.title('Error of transit time between actual data and machine learning')
plt.xlabel('datapoint')
plt.ylabel('absolute error')
plt.savefig('error of transit time actual data vs ml scatter.png')
```



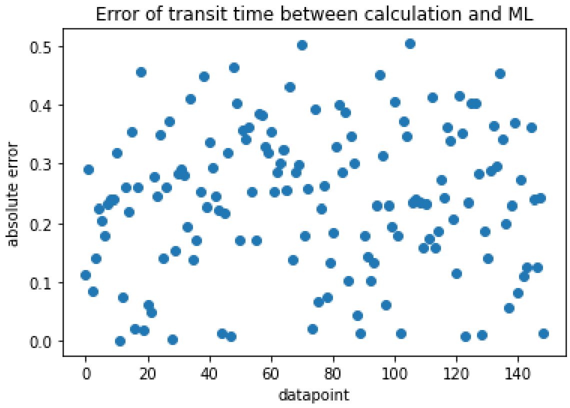
In [21]:
```
plt.hist(dttmd)
plt.title('Error of transit time between actual data and machine learning')
plt.ylabel('total datapoint')
plt.xlabel('absolute error')
plt.savefig('transit time actual data vs ml hist.png')
```



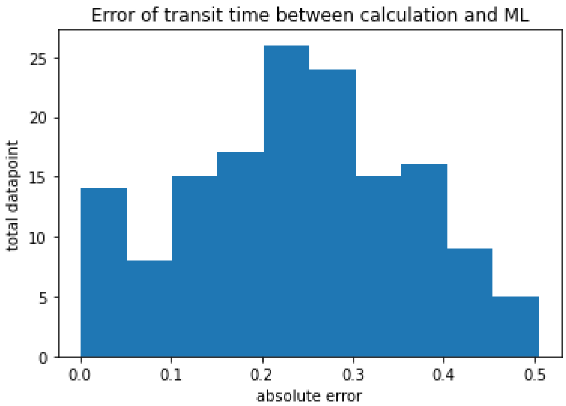
In [22]:
```
plt.scatter(DBM.index,dv1md)
plt.title('Error of impact speed between actual data and machine learning')
plt.xlabel('datapoint')
plt.ylabel('absolute error')
plt.savefig('impact speed error actual data vs ml scatter.png')
```



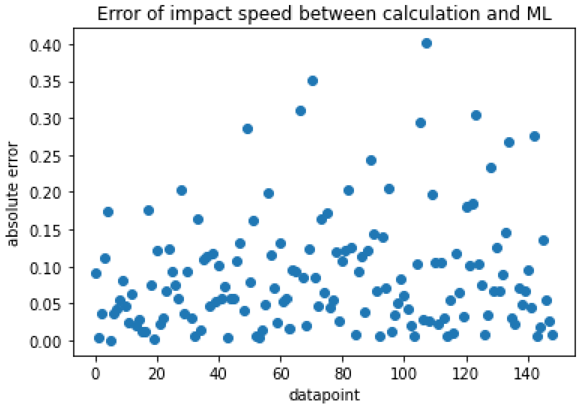
In [23]:
```
plt.scatter(DBM.index,dttmc )
plt.title('Error of transit time between calculation and ML')
plt.xlabel('datapoint')
plt.ylabel('absolute error')
plt.savefig('transit time error calc vs ml scatter.png')
```



In [24]:
```
plt.hist(dttmc)
plt.title('Error of transit time between calculation and ML')
plt.ylabel('total datapoint')
plt.xlabel('absolute error')
plt.savefig('transit time error calc vs ml hist.png')
```

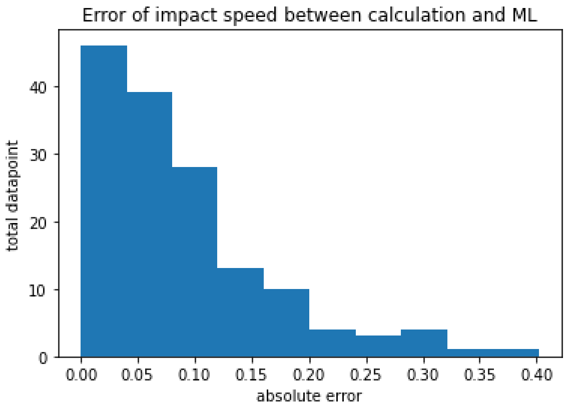


In [25]:
```
plt.scatter(DBM.index,dv1mc)
plt.title('Error of impact speed between calculation and ML')
plt.xlabel('datapoint')
plt.ylabel('absolute error')
plt.savefig('impact speed error calc vs ml scatter.png')
```



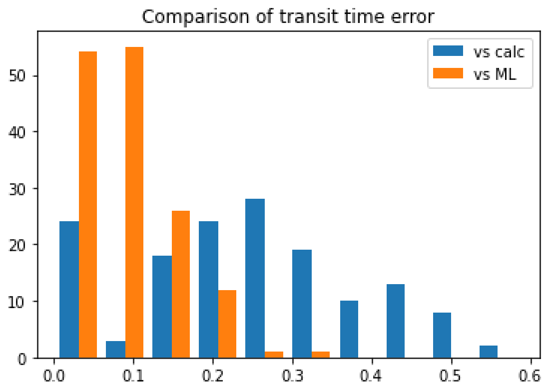
In [26]:

```
plt.hist(dv1mc)
plt.title('Error of impact speed between calculation and ML')
plt.ylabel('total datapoint')
plt.xlabel('absolute error')
plt.savefig('impact speed error calc vs ml hist.png')
```



In [30]:

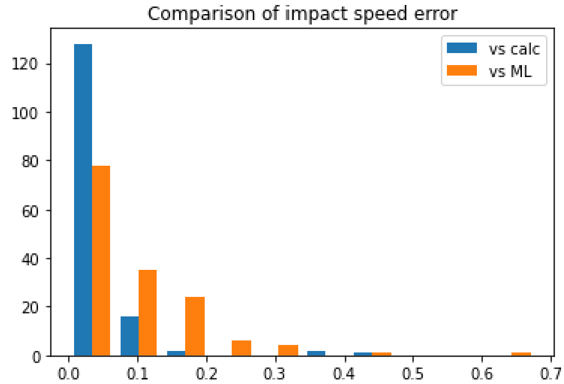
```
plt.hist([dttcd, dttmd], label=['vs calc', 'vs ML'])
plt.legend(loc='upper right')
plt.title('Comparison of transit time error')
plt.show()
plt.savefig('comparison of transit time error.png')
```




```
<Figure size 432x288 with 0 Axes>
```

In [31]:

```
plt.hist([dv1cd, dv1md], label=['vs calc', 'vs ML'])
plt.legend(loc='upper right')
plt.title('Comparison of impact speed error')
plt.show()
plt.savefig('comparison of impact speed error.png')
```




```
<Figure size 432x288 with 0 Axes>
```

In [29]:

```
r2_tt = r2_score(tt_calc, DBM['tt_ml'])
r2_v1 = r2_score(v1_calc, DBM['v1_ml'])
r2_tt, r2_v1
```

Out[29]:

```
(0.3534264714891522, 0.880595510671019)
```