preprocessing

October 16, 2019

In [16]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

In [17]:
```
df = pd.read_csv('db.csv')
```

In [18]:
```
df.head()
```

Out[18]:

	Date/Time - Peak Brightness (UT)	Latitude (Deg)	Longitude (Deg)	\
0	11/21/2009 08:53:00 PM	22.0S	29.2E	
1	01/09/2015 10:41:11 AM	2.0N	28.8E	
2	05/16/2014 12:42:48 PM	44.2S	176.2W	
3	08/23/2014 06:29:41 AM	61.7S	132.6E	
4	12/12/2014 06:48:11 AM	33.5N	144.9E	

	Altitude (km)	Velocity (km/s)	Velocity Components (km/s): vx	\
0	38.0	32.1	3.0	
1	36.0	NaN	-10.7	
2	44.0	NaN	14.4	
3	22.2	16.2	-2.3	
4	26.3	NaN	11.5	

	Velocity Components (km/s): vy	Velocity Components (km/s): vz	\
0	-17.0	-27.0	
1	-7.6	11.6	
2	4.6	6.5	
3	5.7	16.5	
4	-2.8	-2.2	

	Total Radiated Energy (J)	Calculated Total Impact Energy (kt)
0	10000000000000	18.00
1	139000000000	0.41
2	309000000000	0.82

```
3              3820000000000                                    7.60
4                33000000000                                    0.11
```

In [19]:
```
# rename colums
df = df.rename(columns= {'Date/Time - Peak Brightness (UT)':'date',
                         'Latitude (Deg)':'lat',
                         'Longitude (Deg)':'long',
                        'Altitude (km)': 'alt',
                         'Velocity (km/s)': 'vel',
                         'Velocity Components (km/s): vx': 'velx',
                         'Velocity Components (km/s): vy': 'vely',
                         'Velocity Components (km/s): vz': 'velz',
                         'Total Radiated Energy (J)':'tot_j',
                         'Calculated Total Impact Energy (kt)': 'tot_kt'
                        })

r, c = df.shape
```

In [20]:
```
## missing data

missing_data = pd.DataFrame({'total_missing': df.isnull().sum(), 'perc_missing': (df.
missing_data
```

Out[20]:

	perc_missing	total_missing
date	0.000000	0
lat	0.000000	0
long	0.000000	0
alt	26.086957	24
vel	92.391304	85
velx	43.478261	40
vely	43.478261	40
velz	43.478261	40
tot_j	0.000000	0
tot_kt	0.000000	0

In [21]:
```
df.describe()
```

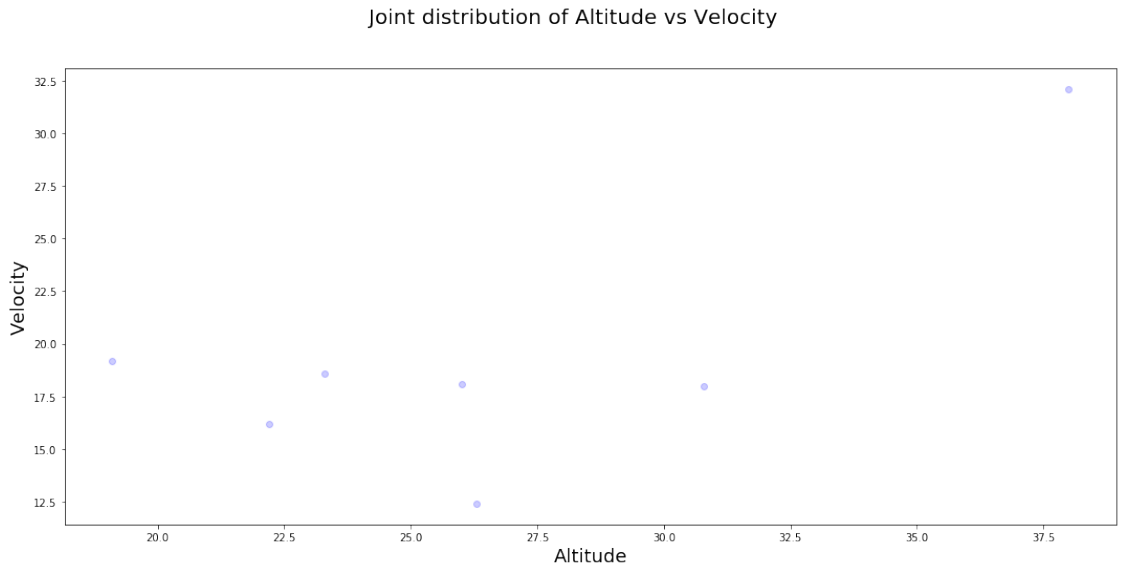
Out[21]:

	alt	vel	velx	vely	velz	tot_j	\
count	68.000000	7.000000	52.000000	52.000000	52.000000	9.200000e+01	
mean	32.314706	19.228571	1.469231	-2.836538	-1.138462	4.897685e+12	
std	8.613219	6.119018	11.427033	11.861175	10.081144	3.913597e+13	
min	18.700000	12.400000	-35.400000	-43.500000	-27.000000	2.000000e+10	
25%	26.300000	17.100000	-5.050000	-11.725000	-7.925000	3.875000e+10	
50%	30.700000	18.100000	1.850000	-2.300000	-1.600000	7.400000e+10	
75%	37.000000	18.900000	10.375000	4.825000	5.600000	1.905000e+11	
max	66.600000	32.100000	21.300000	16.100000	17.000000	3.750000e+14	

	tot_kt
count	92.000000

```
mean     6.347348
std     46.010305
min      0.073000
25%      0.130000
50%      0.230000
75%      0.535000
max    440.000000
```

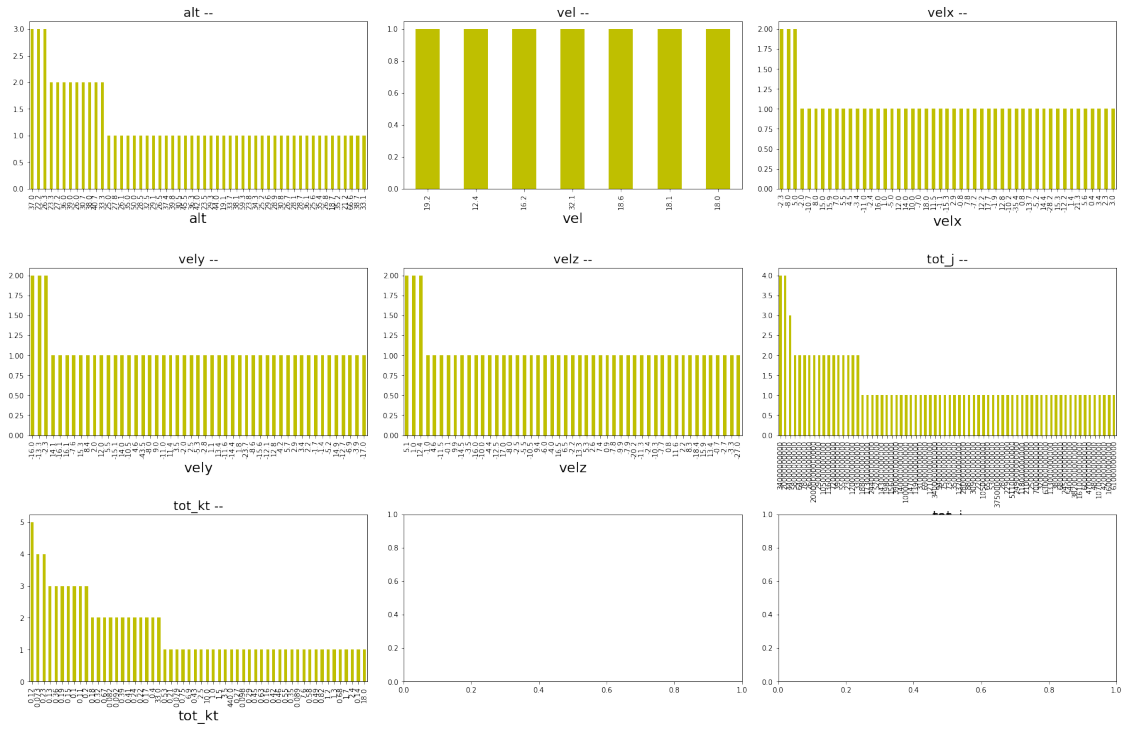
In [22]:
```
plt.figure(figsize=(18,8))
plt.xlabel("Altitude", fontsize=18)
plt.ylabel("Velocity", fontsize=18)
plt.suptitle("Joint distribution of Altitude vs Velocity", fontsize= 20)
plt.plot(df['alt'], df['vel'], 'bo', alpha=0.2)
plt.show()
```



In [24]:
```
def plot_dist(col, ax):
    df[col][df[col].notnull()].value_counts().plot('bar', facecolor='y', ax=ax)
    ax.set_xlabel('{}'.format(col), fontsize=20)
    ax.set_title("{} --".format(col), fontsize= 18)
    return ax

f, ax = plt.subplots(3,3, figsize = (22,15))
f.tight_layout(h_pad=9, w_pad=2, rect=[0, 0.03, 1, 0.93])
cols = ['alt','vel', 'velx', 'vely','velz', 'tot_j', 'tot_kt']
k = 0
for i in range(3):
    for j in range(3):
        plot_dist(cols[k], ax[i][j])
```

```
            k += 1
    __ = plt.suptitle("Initial Distributions of features", fontsize= 25)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)

<ipython-input-24-1f4b8d1a0d9f> in <module>()
     11 for i in range(3):
     12     for j in range(3):
---> 13         plot_dist(cols[k], ax[i][j])
     14         k += 1
     15 __ = plt.suptitle("Initial Distributions of features", fontsize= 25)

IndexError: list index out of range
```



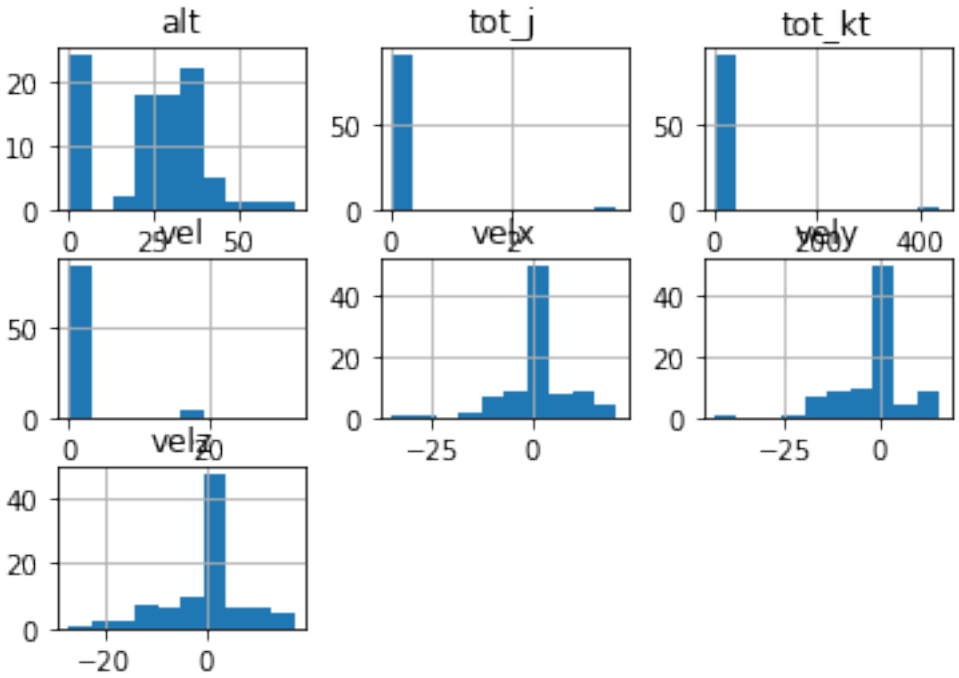

```
In [27]: def plot_barh(df,col, cmap = None, stacked=False, norm = None):
             df.plot(kind='barh', colormap=cmap, stacked=stacked)
             fig = plt.gcf()
             fig.set_size_inches(24,12)
             plt.title("Category vs {}-feedback -  Modcloth {}".format(col, '(Normalized)' if
```

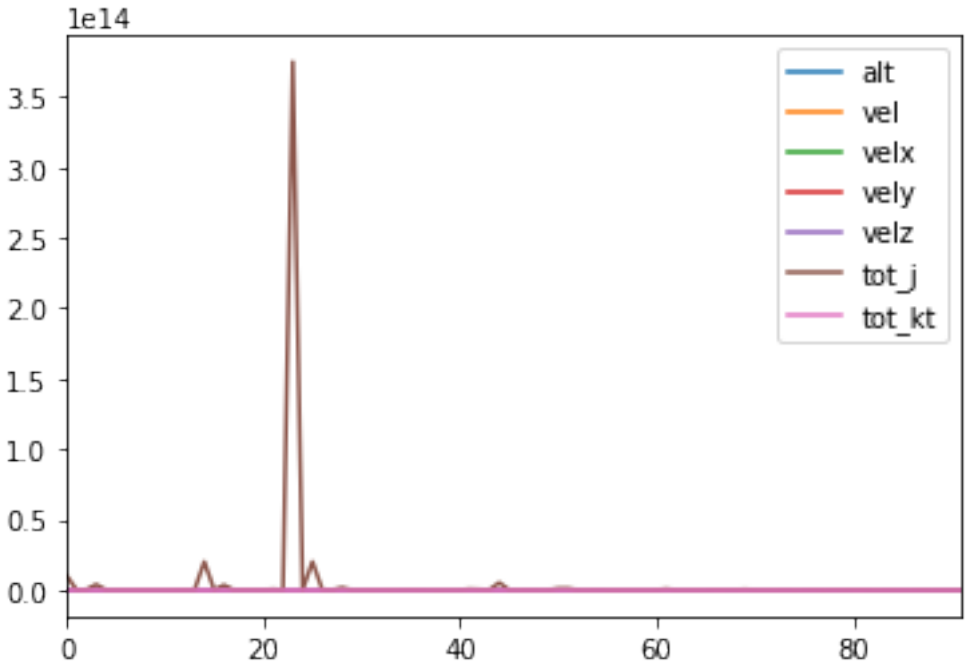
```
        plt.ylabel('Category', fontsize = 18)
        plot = plt.xlabel('Frequency', fontsize=18)

    def norm_counts(t):
        norms = np.linalg.norm(t.fillna(0), axis=1)
        t_norm = t[0:0]
        for row, euc in zip(t.iterrows(), norms):
            t_norm.loc[row[0]] = list(map(lambda x: x/euc, list(row[1])))
        return t_norm
```

In [29]:
```
to_drop = ['date', 'lat', 'long']

df = df.drop(to_drop, axis=1)
df = df.replace(np.nan, 0)
df.head()
X = df.iloc[:, :-1].values
Y = df.iloc[:, -1].values
```

In [30]:
```
df.hist()
df.plot()
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x11856d150>





In [31]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=
```

In [32]:
```
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

In [ ]: