In [4]:

```
import dask.dataframe as dd
from dask.diagnostics import ProgressBar
ProgressBar().register()
import multiprocessing
nCPU = multiprocessing.cpu_count() # no. of division/CPUs for parallel d
import pandas as pd
pd.options.mode.chained_assignment = None
from tqdm import tqdm_notebook as tqdm
import os
import re
from datetime import date, datetime, timedelta, timezone
import time
import numpy as np
import itertools
from statistics import mean, median, variance, stdev
import seaborn as sns
from sklearn.cluster import DBSCAN
import json
import matplotlib.pyplot as plt
import collections
import ast
import pickle
import holidays
us_holidays = holidays.UnitedStates()
```

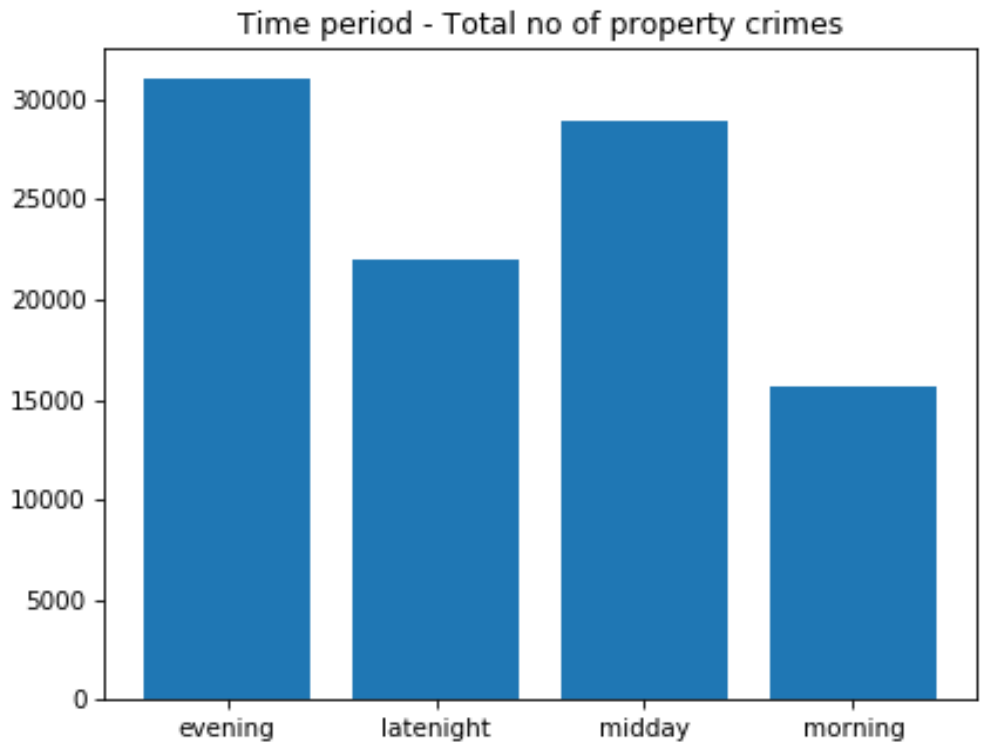
os
osax

```
/Users/koheiyamamoto/.pyenv/versions/anaconda3-5.3.1/lib/python3.7/site-packages/d
ask/dataframe/utils.py:13: FutureWarning: pandas.util.testing is deprecated. Use t
he functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

Temporal Cycles

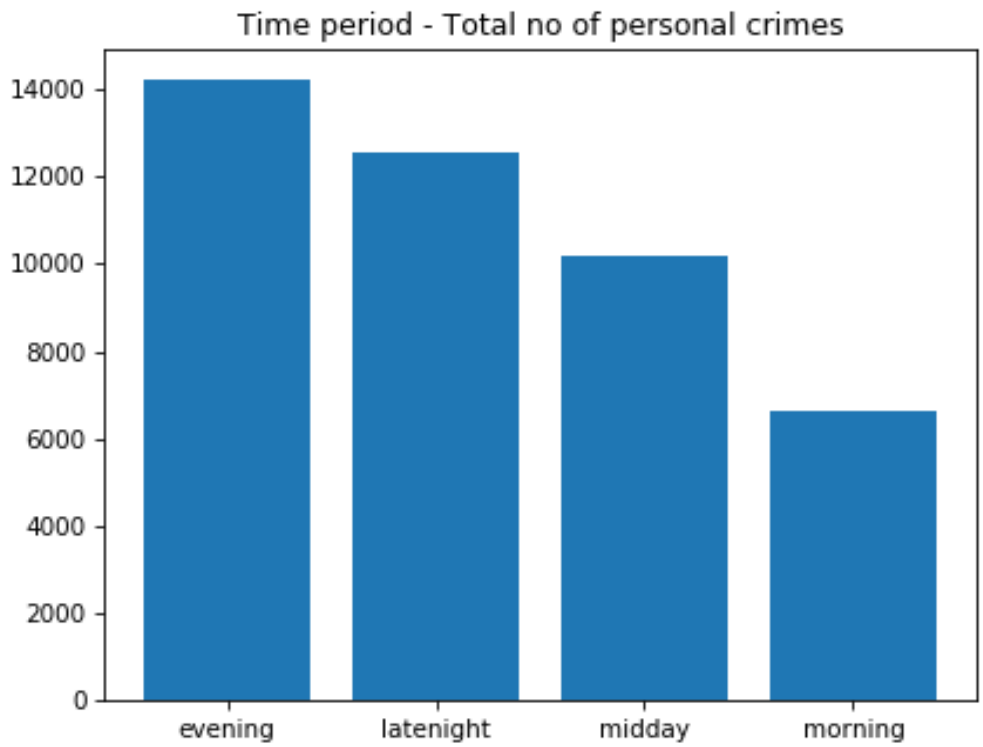
In [303]:

```
crime_property = pd.read_csv('1property_near_sorted_count.csv')
crime_property = crime_property[crime_property['Date'] != '(blank)']
crime_personal = pd.read_csv('1personal_near_sorted_count.csv')
crime_both = pd.concat([crime_property, crime_personal])

timep_volume = pd.DataFrame(crime_property.groupby(['Time_Period', 'Volu
timep_volume['sum'] = timep_volume['Volume'] * timep_volume[0]
x = timep_volume.Time_Period.unique().tolist()
y = [sum(timep_volume.query('Time_Period == @i')['sum'].tolist()) for i

%matplotlib notebook
plt.bar(x, y)
plt.title('Time period - Total no of property crimes')
plt.show()
plt.savefig('res/1_timep_propertycrimes.png')
```



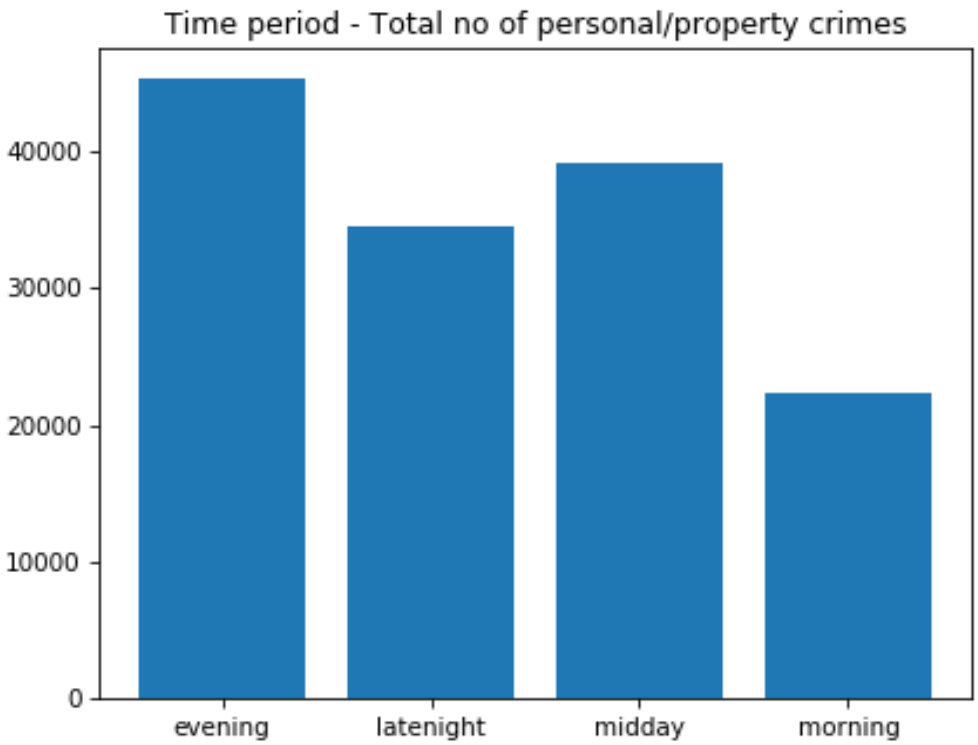
In [333]:

```
timep_volume = pd.DataFrame(crime_personal.groupby(['Time_Period', 'Volu
timep_volume['sum'] = timep_volume['Volume'] * timep_volume[0]
x = timep_volume.Time_Period.unique().tolist()
y = [sum(timep_volume.query('Time_Period == @i')['sum'].tolist()) for i

%matplotlib notebook
plt.bar(x, y)
plt.title('Time period - Total no of personal crimes')
plt.show()
plt.savefig('res/2_timep_personalcrimes.png')
```



In [304]:

```
timep_volume = pd.DataFrame(crime_both.groupby(['Time_Period', 'Volume']
timep_volume['sum'] = timep_volume['Volume'] * timep_volume[0]
x = timep_volume.Time_Period.unique().tolist()
y = [sum(timep_volume.query('Time_Period == @i')['sum'].tolist()) for i

%matplotlib notebook
plt.bar(x, y)
plt.title('Time period - Total no of personal/property crimes')
plt.show()
plt.savefig('res/3_timep_both.png')
```

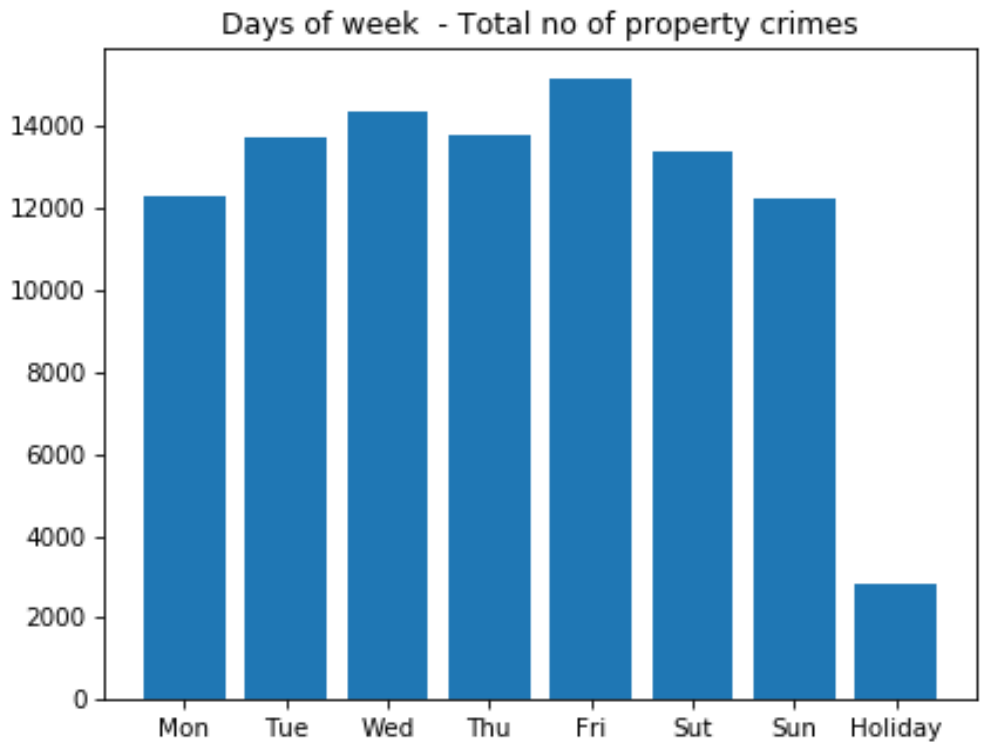


Temporal: weekday +

weekend/holiday

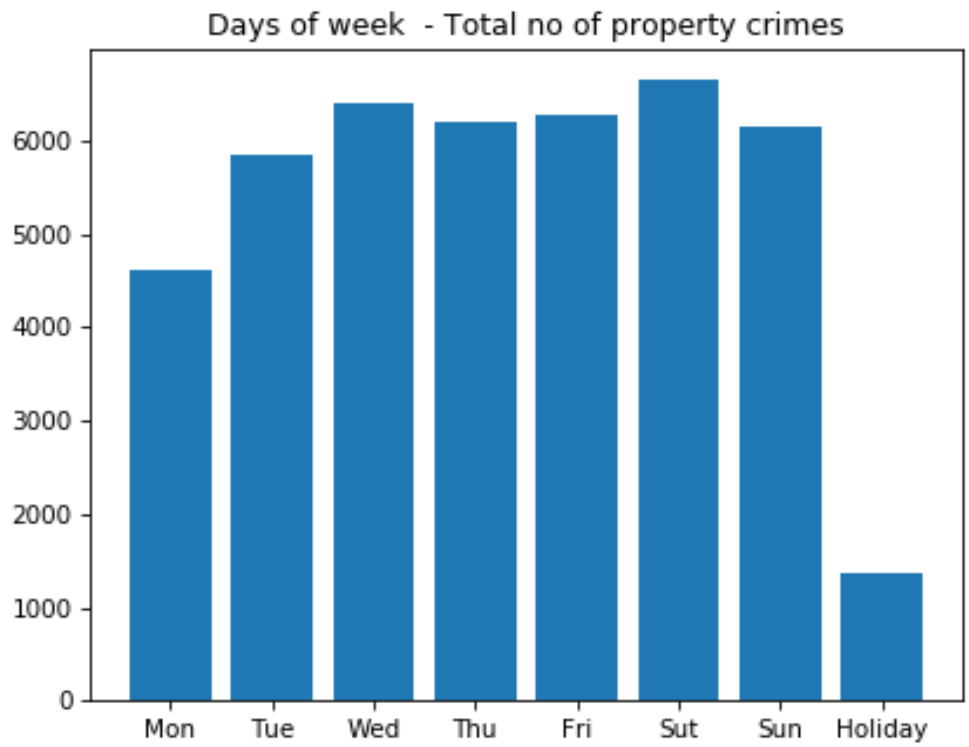
In [332]:

```
def get_weekno(_Date):
    return 7 if datetime.strptime(_Date, '%Y-%m-%d') in us_holidays else

crime_property['weekno'] = crime_property['Date'].apply(get_weekno)
daykind_volume = pd.DataFrame(crime_property.groupby(['weekno', 'Volume'
daykind_volume['sum'] = daykind_volume['Volume'] * daykind_volume[0]
x = daykind_volume.weekno.unique().tolist()
y = [sum(daykind_volume.query('weekno == @i')['sum'].tolist()) for i in

%matplotlib notebook
plt.bar(['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sut', 'Sun', 'Holiday'], y)
plt.title('Days of week  - Total no of property crimes')
plt.show()
plt.savefig('res/4_daykind_propertycrimes.png')
```



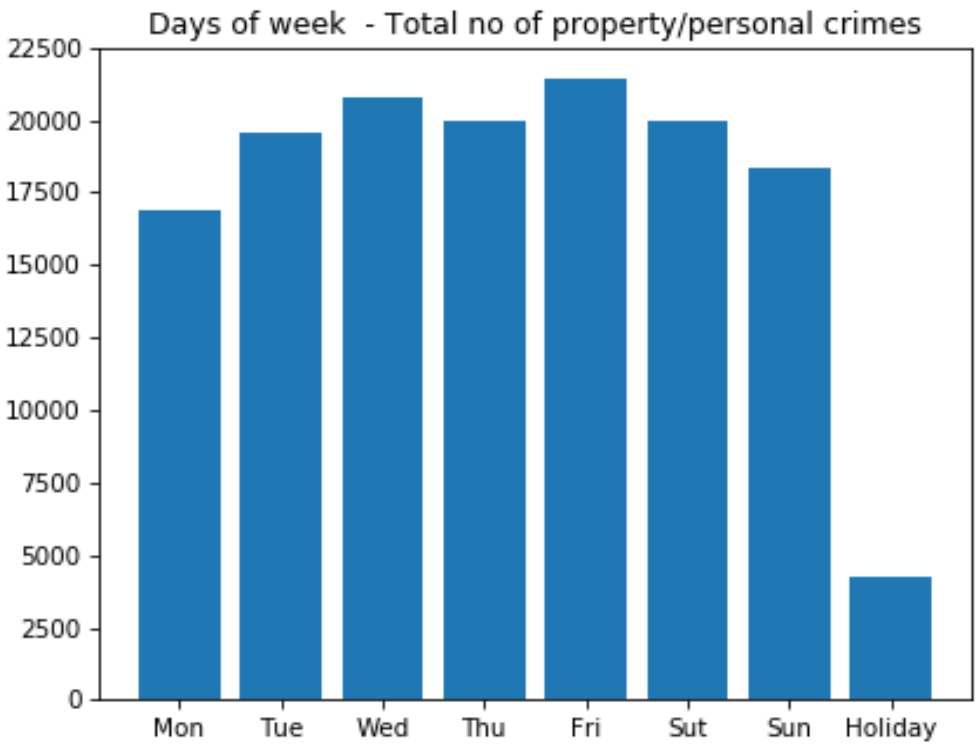
In [137]:

```
def get_weekno(_Date):
    return 7 if datetime.strptime(_Date, '%Y-%m-%d') in us_holidays else

crime_personal['weekno'] = crime_personal['Date'].apply(get_weekno)
daykind_volume = pd.DataFrame(crime_personal.groupby(['weekno', 'Volume'
daykind_volume['sum'] = daykind_volume['Volume'] * daykind_volume[0]
x = daykind_volume.weekno.unique().tolist()
y = [sum(daykind_volume.query('weekno == @i')['sum'].tolist()) for i in

%matplotlib notebook
plt.bar(['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sut', 'Sun', 'Holiday'], y)
plt.title('Days of week  - Total no of property crimes')
plt.show()
plt.savefig('res/5_daykind_personalcrimes.png')
```



In [305]:

```
def get_weekno(_Date):
    return 7 if datetime.strptime(_Date, '%Y-%m-%d') in us_holidays else

crime_both['weekno'] = crime_both['Date'].apply(get_weekno)
daykind_volume = pd.DataFrame(crime_both.groupby(['weekno', 'Volume']).s
daykind_volume['sum'] = daykind_volume['Volume'] * daykind_volume[0]
x = daykind_volume.weekno.unique().tolist()
y = [sum(daykind_volume.query('weekno == @i')['sum'].tolist()) for i in

%matplotlib notebook
plt.bar(['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sut', 'Sun', 'Holiday'], y)
plt.title('Days of week  - Total no of property/personal crimes')
plt.show()
plt.savefig('res/6_daykind_both.png')
```



Some different patterns are obsrved among property, personal and mixed crimes hereby analysis is going to be done with these three differetn dataset

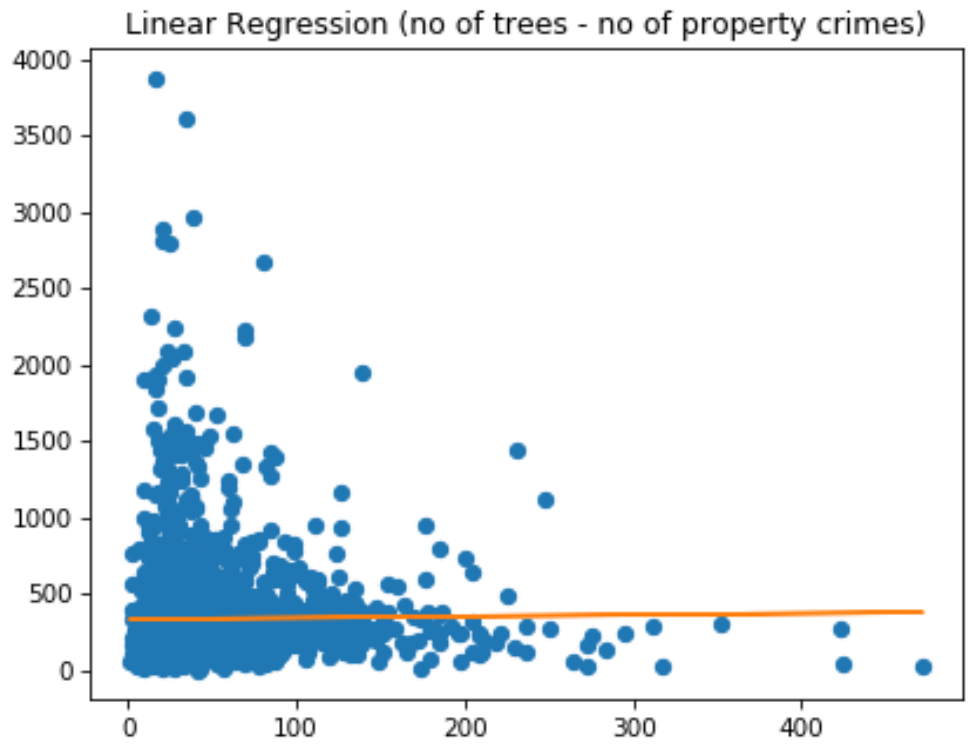
Risk-Terrain - Linear Regression

In [306]:

```
crime_property = pd.read_csv('1property_near_sorted_count.csv')
crime_property = crime_property[crime_property['Date'] != '(blank)']
crime_personal = pd.read_csv('1personal_near_sorted_count.csv')
crime_both = pd.concat([crime_property, crime_personal])

crime_property_census = pd.DataFrame(crime_property.groupby(['Census_Tra
crime_property_census['sum'] = crime_property_census['Volume'] * crime_p
crime_property_census = crime_property_census.groupby('Census_Tracts').a
crime_personal_census = pd.DataFrame(crime_personal.groupby(['Census_Tra
crime_personal_census['sum'] = crime_personal_census['Volume'] * crime_p
crime_personal_census = crime_personal_census.groupby('Census_Tracts').a
crime_both_census = pd.DataFrame(crime_both.groupby(['Census_Tracts', 'V
crime_both_census['sum'] = crime_both_census['Volume'] * crime_both_cens
crime_both_census = crime_both_census.groupby('Census_Tracts').apply(lam

tree = pd.read_csv('7tree_num.csv')
house = pd.read_csv('6housing_density.csv')
population_income = pd.read_csv('45population_income.csv')

crime_property_census = pd.merge(crime_property_census, tree, left_on='C
crime_property_census = pd.merge(crime_property_census, house, left_on='
crime_property_census = pd.merge(crime_property_census, population_incom
crime_personal_census = pd.merge(crime_personal_census, tree, left_on='C
crime_personal_census = pd.merge(crime_personal_census, house, left_on='
crime_personal_census = pd.merge(crime_personal_census, population_incom
crime_both_census = pd.merge(crime_both_census, tree, left_on='Census_Tr
crime_both_census = pd.merge(crime_both_census, house, left_on='Census_T
crime_both_census = pd.merge(crime_both_census, population_income, left_

dropField = ['CTID', 'ctid', 'CT_ID', 'shape_area', 'building_a', 'Censu
crime_property_census.drop(dropField, inplace=True, axis=1)
crime_property_census.dropna(inplace=True)
crime_personal_census.drop(dropField, inplace=True, axis=1)
crime_personal_census.dropna(inplace=True)
crime_both_census.drop(dropField, inplace=True, axis=1)
crime_both_census.dropna(inplace=True)
```

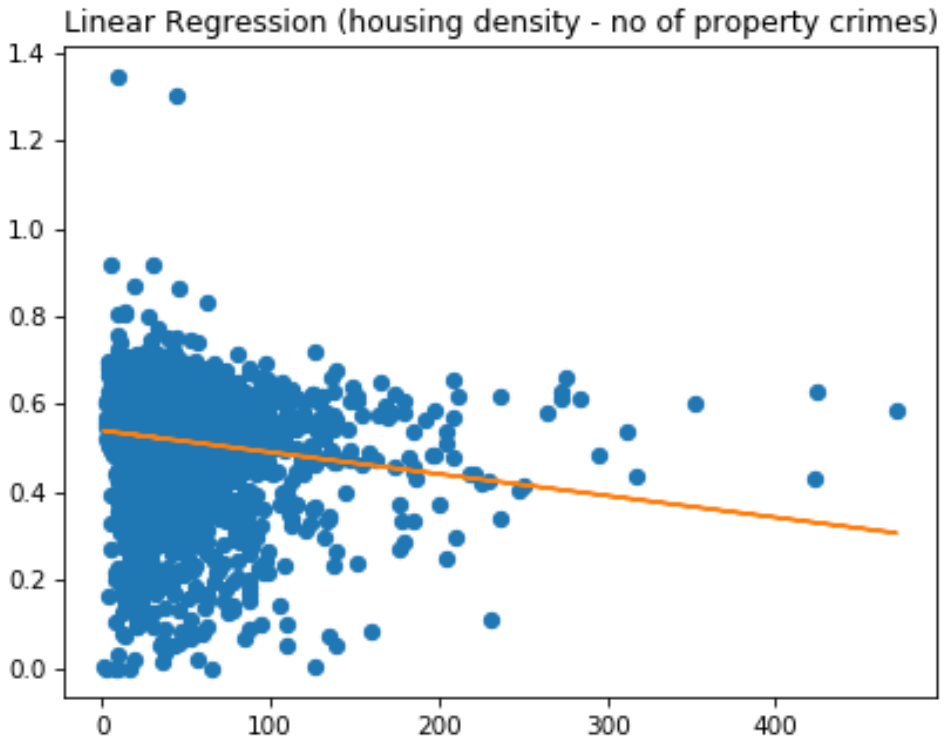
In [311]:

```
y = crime_property_census[['NUMPOINTS']]
x = crime_property_census[['sum']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (no of trees - no of property crimes)')
plt.show()
plt.savefig('res/7_propertycrimes_trees.png')
```



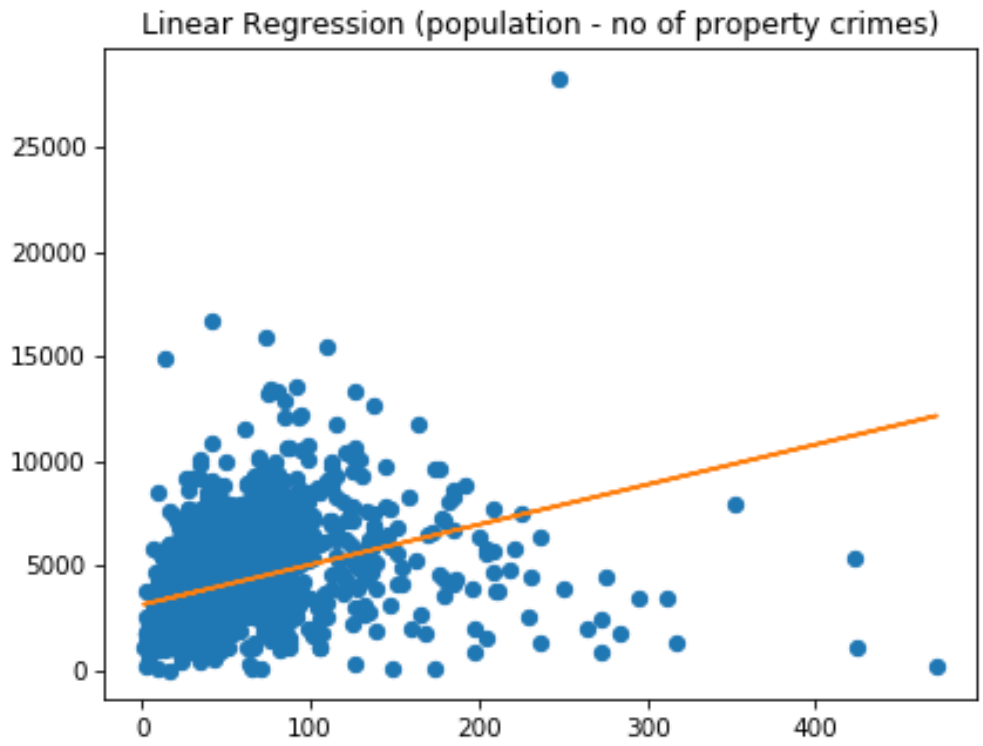
In [312]:

```
y = crime_property_census[['percentage']]
x = crime_property_census[['sum']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (housing density - no of property crimes)')
plt.show()
plt.savefig('res/8_propertycrimes_housing.png')
```



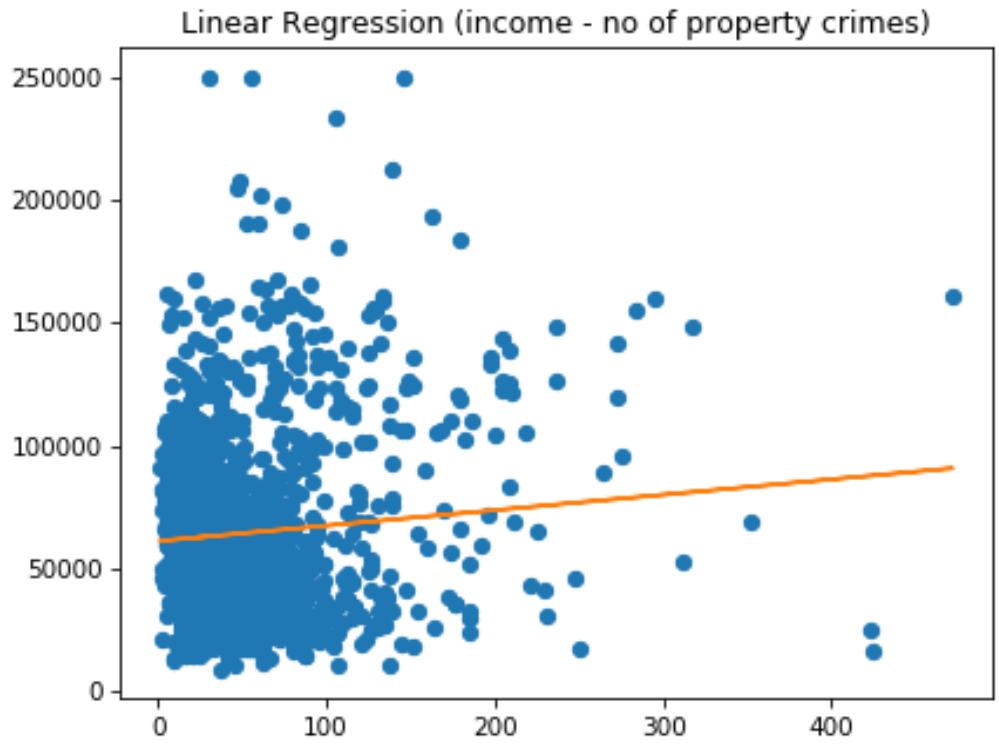
In [313]:

```
y = crime_property_census[['Population']]
x = crime_property_census[['sum']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (population - no of property crimes)')
plt.show()
plt.savefig('res/9_propertycrimes_population.png')
```



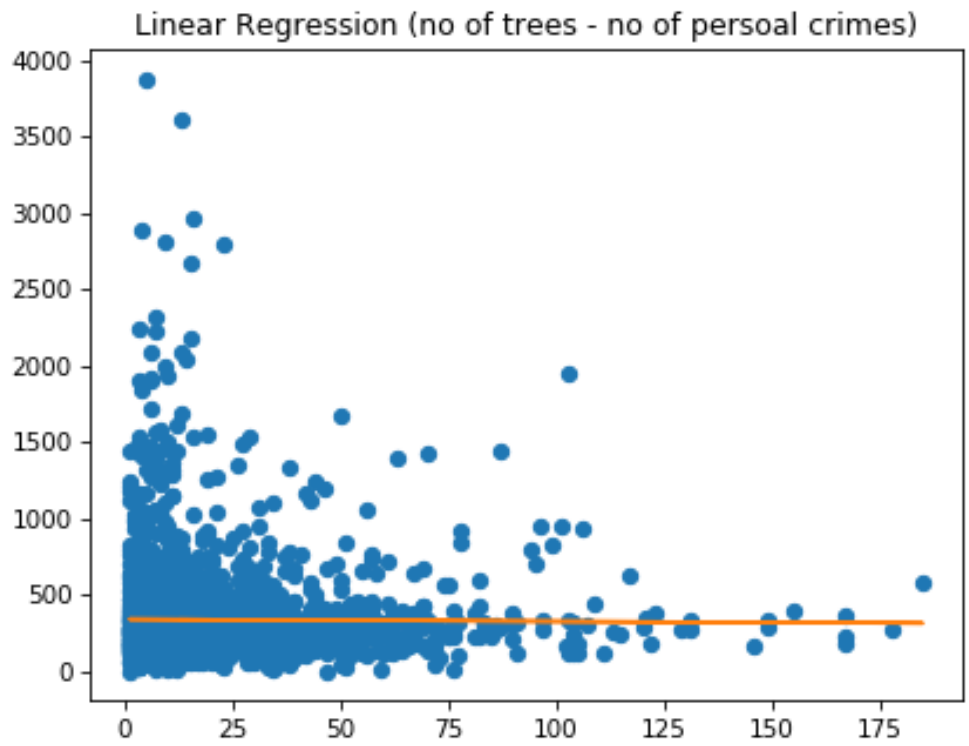
In [314]:

```
y = crime_property_census[['Household_Income']]
x = crime_property_census[['sum']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (income - no of property crimes)')
plt.show()
plt.savefig('res/10_propertycrimes_income.png')
```



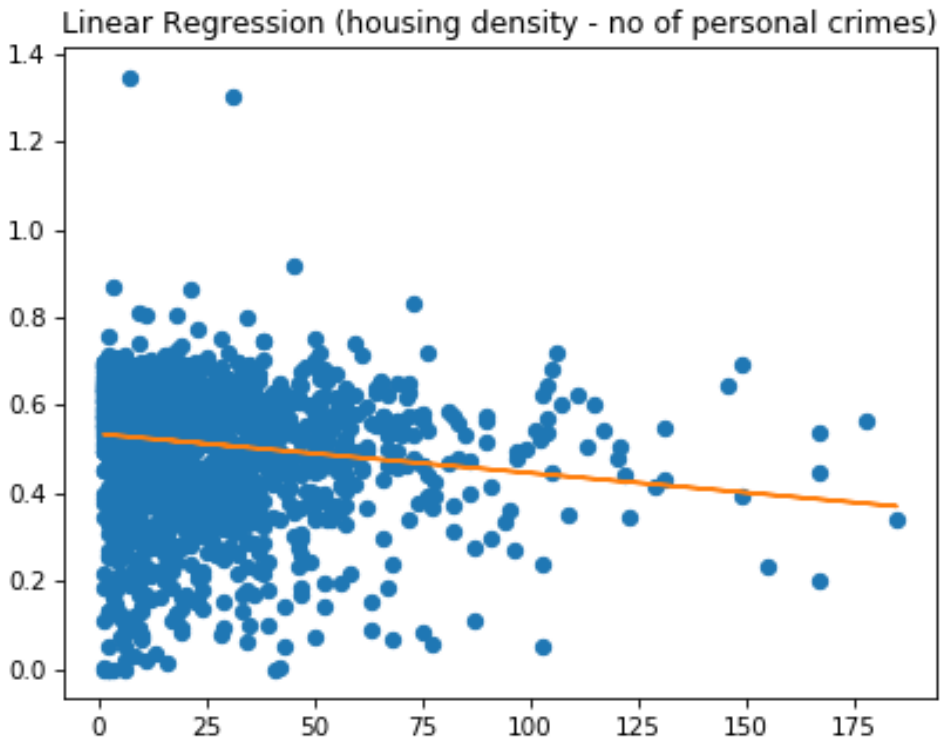
In [315]:

```
y = crime_personal_census[['NUMPOINTS']]
x = crime_personal_census[['sum']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (no of trees - no of persoal crimes)')
plt.show()
plt.savefig('res/11_personalcrimes_trees.png')
```



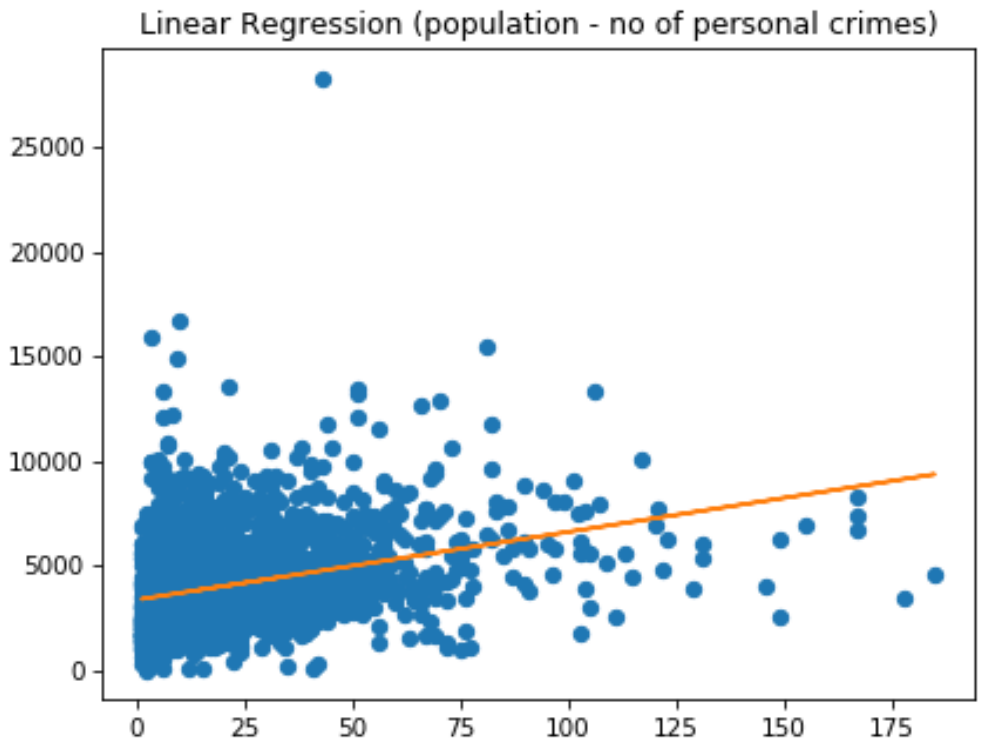
In [316]:

```
y = crime_personal_census[['percentage']]
x = crime_personal_census[['sum']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (housing density - no of personal crimes)')
plt.show()
plt.savefig('res/12_personalcrimes_housing.png')
```



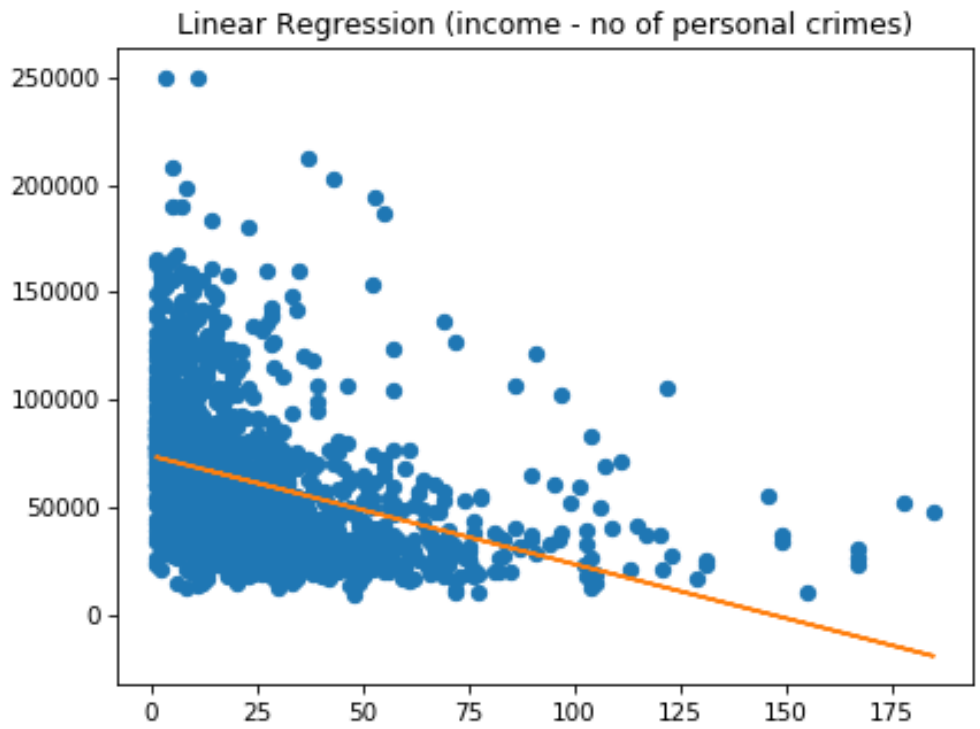
In [317]:

```
y = crime_personal_census[['Population']]
x = crime_personal_census[['sum']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (population - no of personal crimes)')
plt.show()
plt.savefig('res/13_personalcrimes_population.png')
```



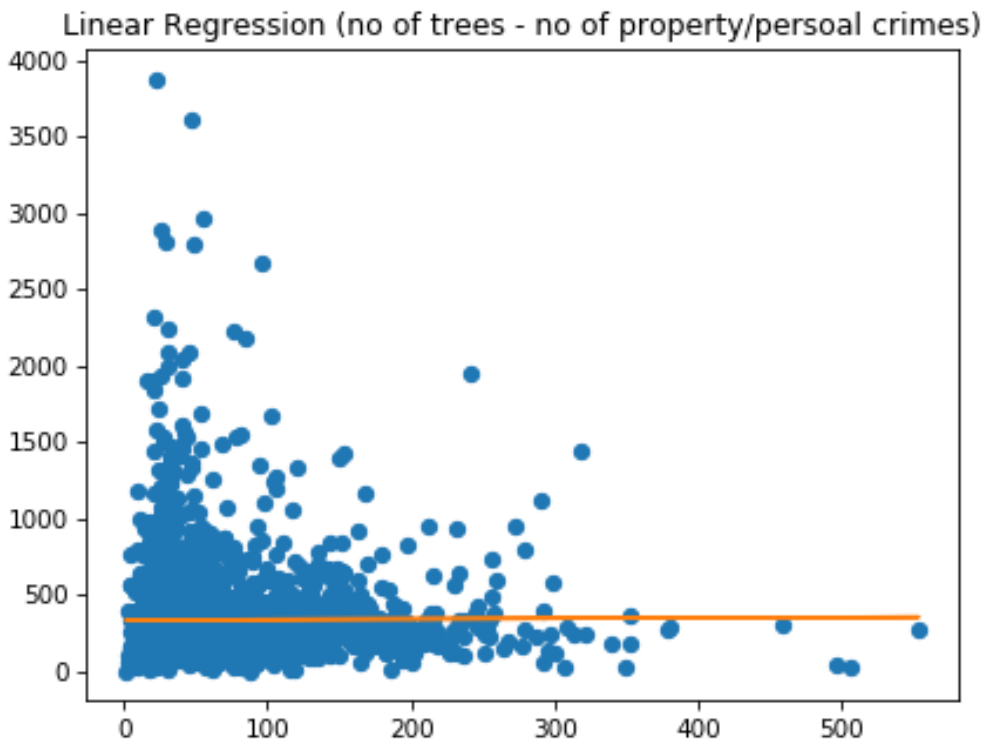
In [318]:

```
y = crime_personal_census[['Household_Income']]
x = crime_personal_census[['sum']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (income - no of personal crimes)')
plt.show()
plt.savefig('res/14_personalcrimes_income.png')
```



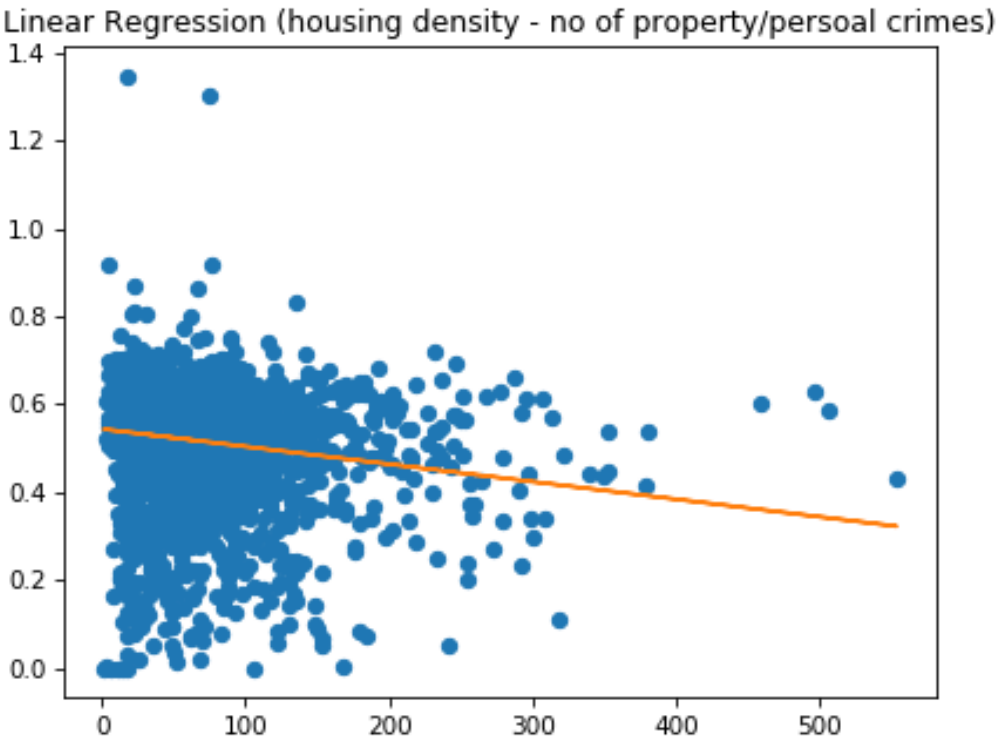
In [319]:

```
y = crime_both_census[['NUMPOINTS']]
x = crime_both_census[['sum']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (no of trees - no of property/persoal crime
plt.show()
plt.savefig('res/15_bothcrimes_trees.png')
```



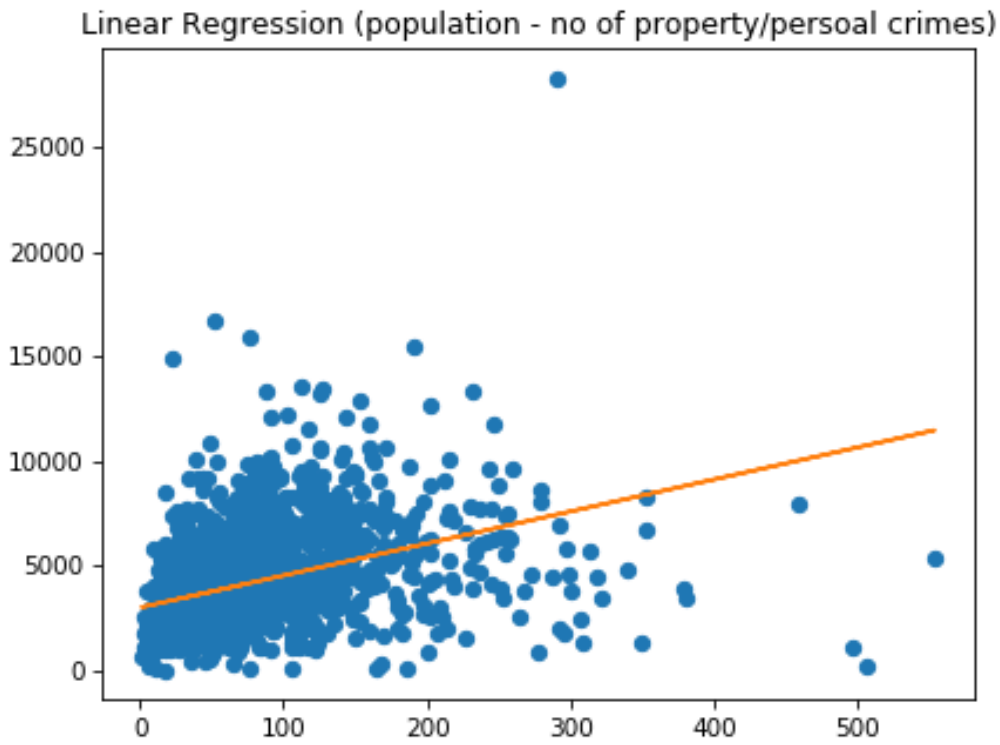
In [320]:

```
y = crime_both_census[['percentage']]
x = crime_both_census[['sum']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (housing density - no of property/persoal c
plt.show()
plt.savefig('res/16_bothcrimes_housing.png')
```



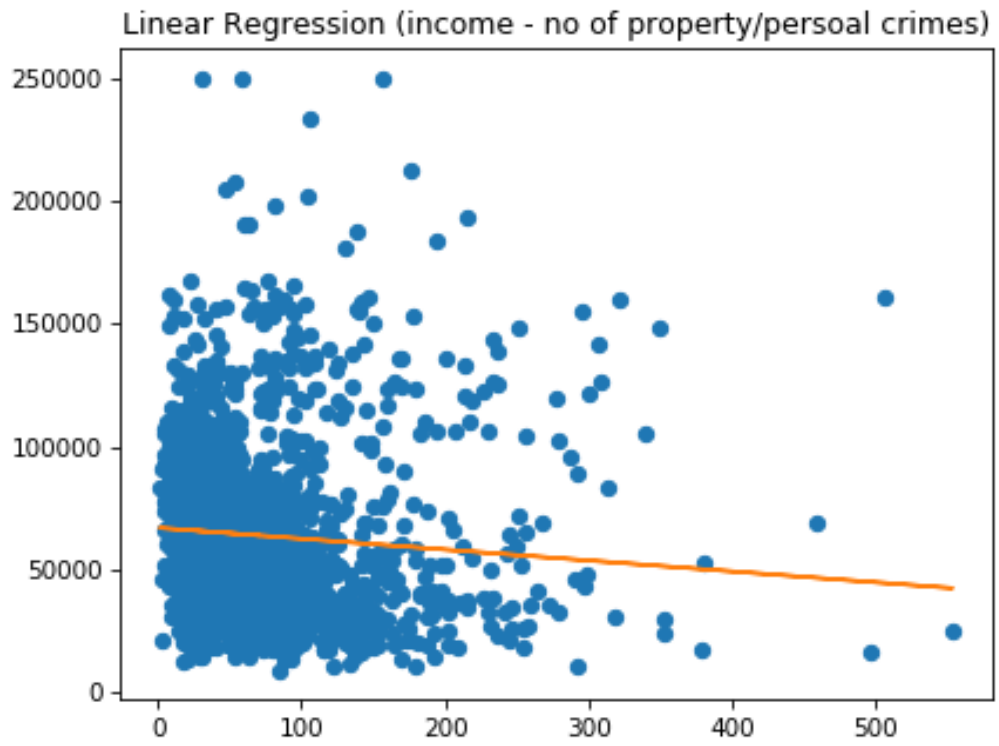
In [321]:

```
y = crime_both_census[['Population']]
x = crime_both_census[['sum']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (population - no of property/persoal crimes
plt.show()
plt.savefig('res/17_bothcrimes_population.png')
```



In [322]:

```
y = crime_both_census[['Household_Income']]
x = crime_both_census[['sum']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (income - no of property/persoal crimes)')
plt.show()
plt.savefig('res/18_bothcrimes_income.png')
```



Multiple regression analysis

In [145]:

```
linear_regression = LinearRegression()
X = crime_property_census.drop("sum", 1)
Y = crime_property_census['sum']
linear_regression.fit(X,Y)
coefficient = pd.DataFrame(X.columns.values)
coefficient['coefficient'] = pd.DataFrame(linear_regression.coef_)
coefficient
```

	0	coefficient
0	NUMPOINTS	-0.020760
1	percentage	-61.110139
2	Population	0.008634
3	Household_Income	0.000201

In [147]:

```
linear_regression = LinearRegression()
X = crime_personal_census.drop("sum", 1)
Y = crime_personal_census['sum']
linear_regression.fit(X,Y)
coefficient = pd.DataFrame(X.columns.values)
coefficient['coefficient'] = pd.DataFrame(linear_regression.coef_)
coefficient
```

	0	coefficient
0	NUMPOINTS	-0.004041
1	percentage	-24.874128
2	Population	0.003604
3	Household_Income	-0.000238

In [323]:

```
linear_regression = LinearRegression()
X = crime_both_census.drop("sum", 1)
Y = crime_both_census['sum']
linear_regression.fit(X,Y)
coefficient = pd.DataFrame(X.columns.values)
coefficient['coefficient'] = pd.DataFrame(linear_regression.coef_)
coefficient
```

	0	coefficient
0	NUMPOINTS	-0.024923
1	percentage	-85.537079
2	Population	0.012302
3	Household_Income	-0.000032

Obviously, linearity is not observed. (Other types of regression may be applied on top of that kernel function causes memory overload given 10000 samples. (this time is not even a problem but Support Vector Regression is not suitable in this sense.))

Random Forest Regression

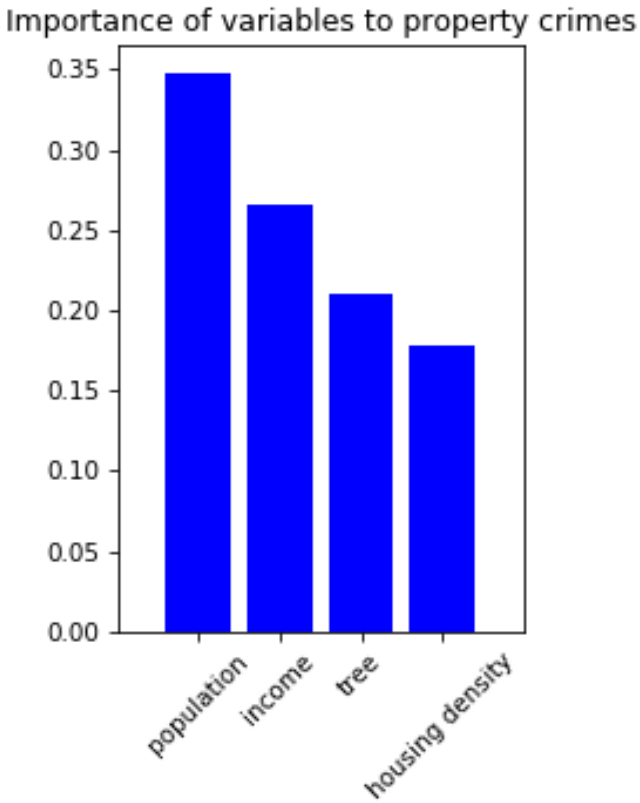
In [549]:

```
crime_property = pd.read_csv('1property_near_sorted_count.csv')
crime_property = crime_property[crime_property['Date'] != '(blank)']
crime_personal = pd.read_csv('1personal_near_sorted_count.csv')
crime_both = pd.concat([crime_property, crime_personal])

crime_property_census = pd.DataFrame(crime_property.groupby(['Census_Tra
crime_property_census['sum'] = crime_property_census['Volume'] * crime_p
crime_property_census = crime_property_census.groupby('Census_Tracts').a
crime_personal_census = pd.DataFrame(crime_personal.groupby(['Census_Tra
crime_personal_census['sum'] = crime_personal_census['Volume'] * crime_p
crime_personal_census = crime_personal_census.groupby('Census_Tracts').a
crime_both_census = pd.DataFrame(crime_both.groupby(['Census_Tracts', 'V
crime_both_census['sum'] = crime_both_census['Volume'] * crime_both_cens
crime_both_census = crime_both_census.groupby('Census_Tracts').apply(lam

tree = pd.read_csv('7tree_num.csv')
house = pd.read_csv('6housing_density.csv')
population_income = pd.read_csv('45population_income.csv')

crime_property_census = pd.merge(crime_property_census, tree, left_on='C
crime_property_census = pd.merge(crime_property_census, house, left_on='
crime_property_census = pd.merge(crime_property_census, population_incom
crime_personal_census = pd.merge(crime_personal_census, tree, left_on='C
crime_personal_census = pd.merge(crime_personal_census, house, left_on='
crime_personal_census = pd.merge(crime_personal_census, population_incom
crime_both_census = pd.merge(crime_both_census, tree, left_on='Census_Tr
crime_both_census = pd.merge(crime_both_census, house, left_on='Census_T
crime_both_census = pd.merge(crime_both_census, population_income, left_

dropField = ['CTID', 'ctid', 'CT_ID', 'shape_area', 'building_a', 'Censu
crime_property_census.drop(dropField, inplace=True, axis=1)
crime_property_census.dropna(inplace=True)
crime_personal_census.drop(dropField, inplace=True, axis=1)
crime_personal_census.dropna(inplace=True)
crime_both_census.drop(dropField, inplace=True, axis=1)
crime_both_census.dropna(inplace=True)
```

In [551]:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
data_test_all = crime_property_census.sample(n=100)
data_train_all = crime_property_census.loc[list(set(crime_property_censu

y_train = data_train_all['sum'].values.tolist()
data_train = data_train_all.drop("sum", 1)
x_train = data_train.values.tolist()

y_test = data_test_all['sum'].values.tolist()
data_test = data_test_all.drop("sum", 1)
x_test = data_test.values.tolist()

rfr = RandomForestRegressor(n_estimators=1000)
rfr.fit(x_train, y_train)
predict_y = rfr.predict(x_test)
r2_score = r2_score(y_test, predict_y)
print('oefficient of determination: ', r2_score)
feature = rfr.feature_importances_
label = data_train.columns[0:]
indices = np.argsort(feature)[::-1]
for i in range(len(feature)):
    print(str(i + 1) + "   " +
          str(label[indices[i]]) + "   " + str(feature[indices[i]]))

%matplotlib notebook
plt.subplot(122, facecolor='white')
plt.title('Importance of variables to property crimes')
plt.bar(
    range(
        len(feature)),
    feature[indices],
    color='blue',
    align='center')
# plt.xticks(range(len(feature)), label[indices], rotation=45)
plt.xticks(range(len(feature)),['population', 'income', 'tree', 'housing
plt.xlim([-1, len(feature)])
plt.tight_layout()
plt.show()
plt.savefig('res/19_property_RF.png')
```

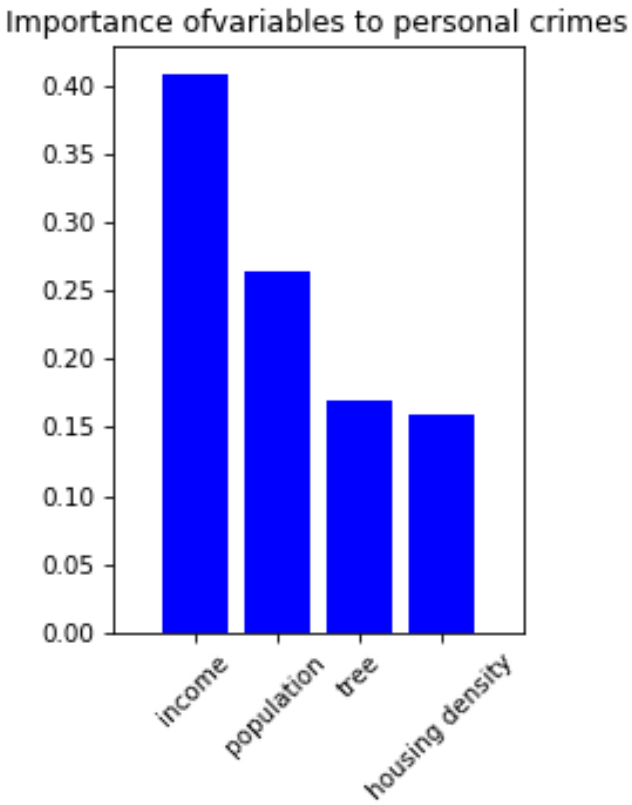
```
oefficient of determination:  0.205595835118758
1   Population   0.3471079548202601
2   Household_Income   0.26593022607222283
3   NUMPOINTS   0.20952571251485202
4   percentage   0.17743610659266523
```



In [554]:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
data_test_all = crime_personal_census.sample(n=100)
data_train_all = crime_personal_census.loc[list(set(crime_personal_censu

y_train = data_train_all['sum'].values.tolist()
data_train = data_train_all.drop("sum", 1)
x_train = data_train.values.tolist()

y_test = data_test_all['sum'].values.tolist()
data_test = data_test_all.drop("sum", 1)
x_test = data_test.values.tolist()

rfr = RandomForestRegressor(n_estimators=1000)
rfr.fit(x_train, y_train)
predict_y = rfr.predict(x_test)
r2_score = r2_score(y_test, predict_y)
print('coefficient of determination: ', r2_score)
feature = rfr.feature_importances_
label = data_train.columns[0:]
indices = np.argsort(feature)[::-1]
for i in range(len(feature)):
    print(str(i + 1) + "   " +
          str(label[indices[i]]) + "   " + str(feature[indices[i]]))

%matplotlib notebook
plt.subplot(122, facecolor='white')
plt.title('Importance ofvariables to personal crimes')
plt.bar(
    range(
        len(feature)),
    feature[indices],
    color='blue',
    align='center')
plt.xticks(range(len(feature)),['income', 'population', 'tree', 'housing
plt.xlim([-1, len(feature)])
plt.tight_layout()
plt.show()
plt.savefig('res/20_personal_RF.png')
```

```
coefficient of determination:  0.46484600300745327
1   Household_Income   0.4075296422460255
2   Population   0.2644270708468366
3   NUMPOINTS   0.16957823543401113
4   percentage   0.15846505147312653
```

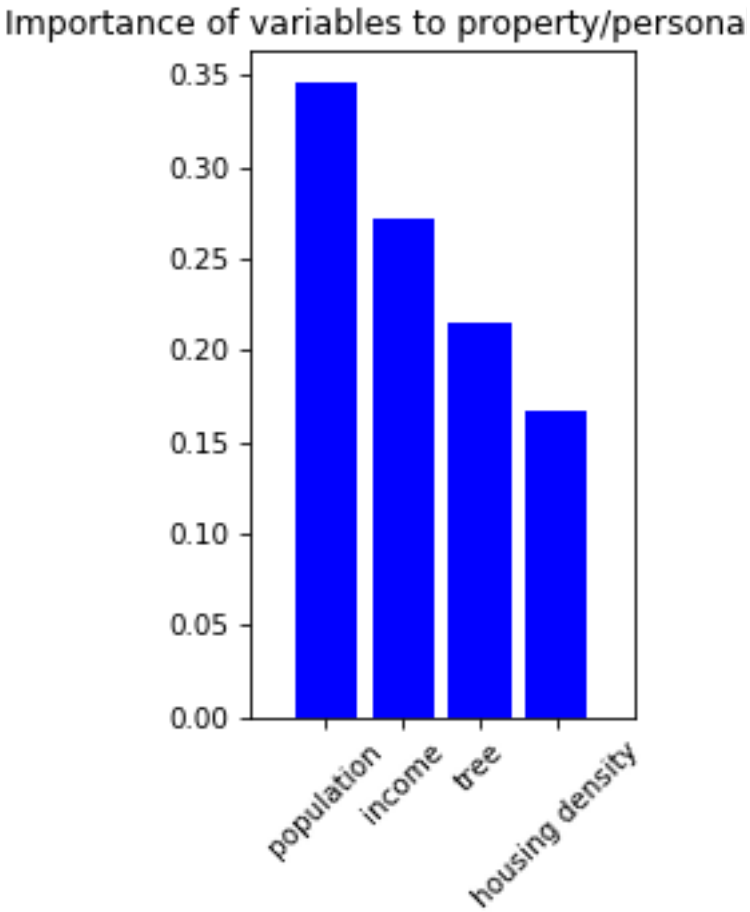


In [556]:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
data_test_all = crime_both_census.sample(n=100)
data_train_all = crime_both_census.loc[list(set(crime_both_census.index.

y_train = data_train_all['sum'].values.tolist()
data_train = data_train_all.drop("sum", 1)
x_train = data_train.values.tolist()

y_test = data_test_all['sum'].values.tolist()
data_test = data_test_all.drop("sum", 1)
x_test = data_test.values.tolist()

rfr = RandomForestRegressor(n_estimators=1000)
rfr.fit(x_train, y_train)
predict_y = rfr.predict(x_test)
r2_score = r2_score(y_test, predict_y)
print('oefficient of determination: ', r2_score)
feature = rfr.feature_importances_
label = data_train.columns[0:]
indices = np.argsort(feature)[::-1]
for i in range(len(feature)):
    print(str(i + 1) + "   " +
          str(label[indices[i]]) + "   " + str(feature[indices[i]]))

%matplotlib notebook
plt.subplot(122, facecolor='white')
plt.title('Importance of variables to property/personal crimes')
plt.bar(
    range(
        len(feature)),
    feature[indices],
    color='blue',
    align='center')
plt.xticks(range(len(feature)),['population', 'income', 'tree', 'housing
plt.xlim([-1, len(feature)])
plt.tight_layout()
plt.show()
plt.savefig('res/21_both_RF.png')
```

```
oefficient of determination:  0.2517983942910358
1   Population   0.34559156049550477
2   Household_Income   0.27248457999455317
3   NUMPOINTS   0.21499578970872207
4   percentage   0.16692806980121994
```

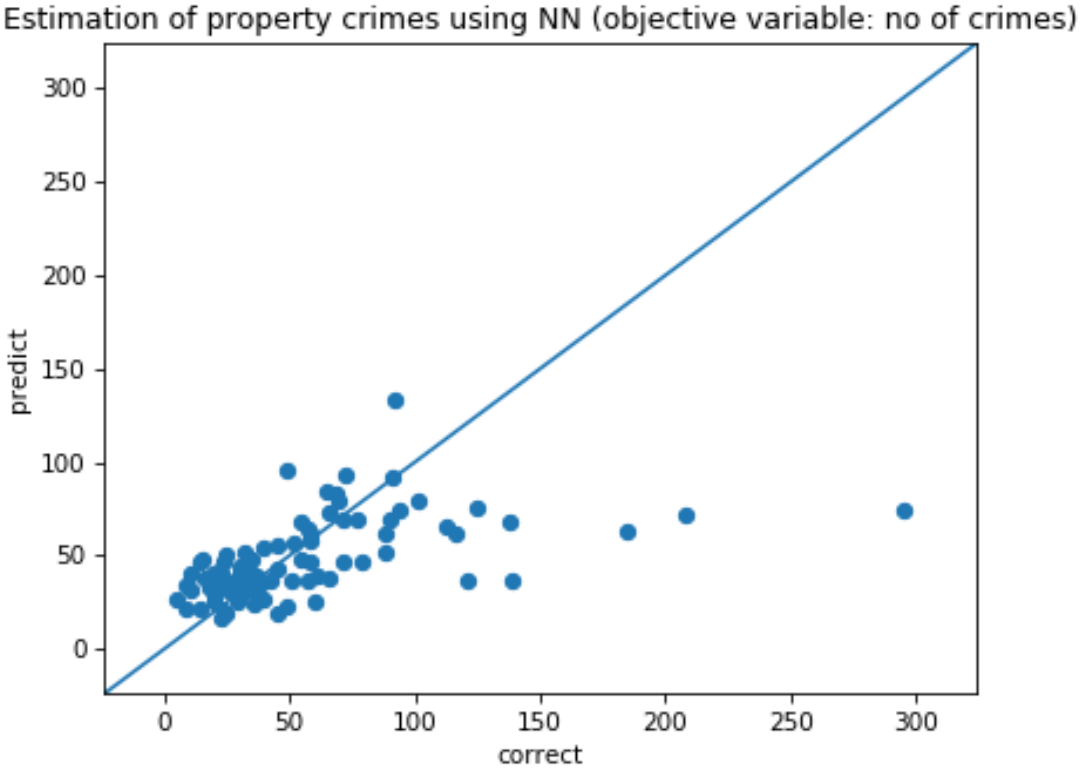
Figure 1



Netural network

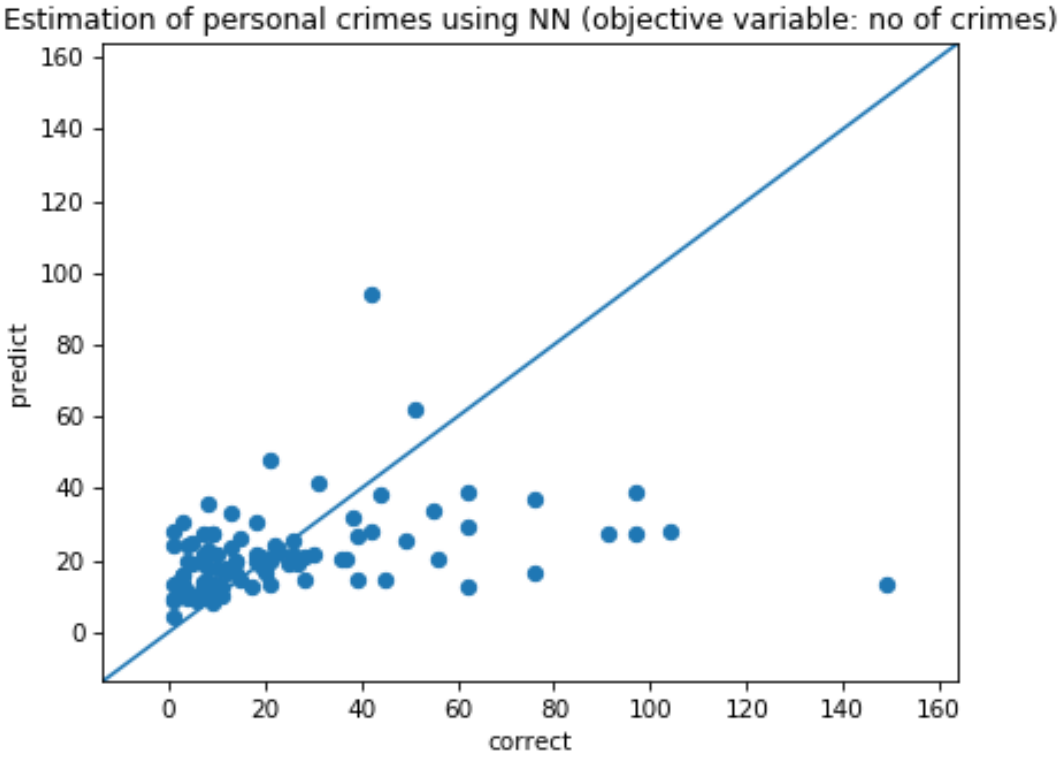
In [345]:

```
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import r2_score

data_test_all = crime_property_census.sample(n=100)
data_train_all = crime_property_census.loc[list(set(crime_property_censu

y_train = data_train_all['sum'].values.tolist()
data_train = data_train_all.drop("sum", 1)
x_train = data_train.values.tolist()

y_test = data_test_all['sum'].values.tolist()
data_test = data_test_all.drop("sum", 1)
x_test = data_test.values.tolist()

mlp = MLPRegressor(hidden_layer_sizes=(100,), max_iter=500) # https://nu
mlp.fit(x_train, y_train)
pred = mlp.predict(x_test)

%matplotlib notebook
values =  np.concatenate([y_test, pred], 0)
ptp = np.ptp(values)
min_value = np.min(values) - ptp * 0.1
max_value = np.max(values) + ptp * 0.1

plt.scatter(y_test, pred)
plt.plot([min_value, max_value], [min_value, max_value])
plt.xlim(min_value, max_value)
plt.ylim(min_value, max_value)
plt.xlabel('correct')
plt.ylabel('predict')
plt.title('Estimation of property crimes using NN (objective variable: n
plt.show()
plt.savefig('res/22_property_NN.png')
```


Estimation of property crimes using NN (objective variable: no of crimes)
predict
correct
0
50
100
150
200
250
300

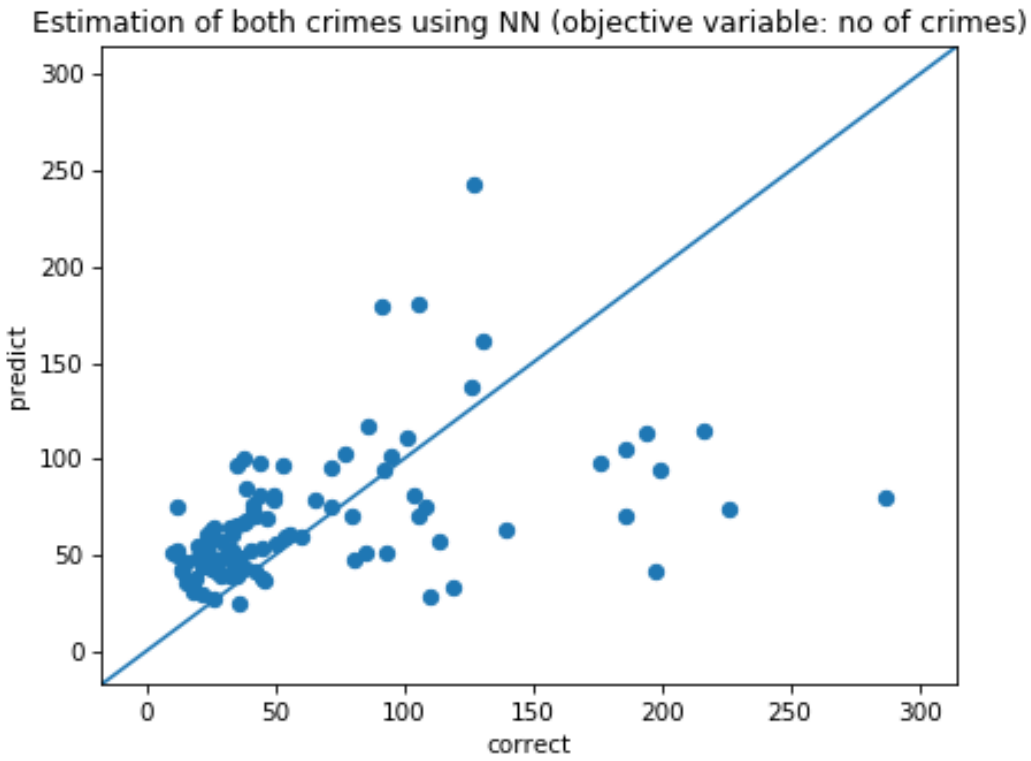
In [347]:

```
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import r2_score

data_test_all = crime_personal_census.sample(n=100)
data_train_all = crime_personal_census.loc[list(set(crime_personal_censu

y_train = data_train_all['sum'].values.tolist()
data_train = data_train_all.drop("sum", 1)
x_train = data_train.values.tolist()

y_test = data_test_all['sum'].values.tolist()
data_test = data_test_all.drop("sum", 1)
x_test = data_test.values.tolist()

mlp = MLPRegressor(hidden_layer_sizes=(100,), max_iter=500) # https://nu
mlp.fit(x_train, y_train)
pred = mlp.predict(x_test)

%matplotlib notebook
values =  np.concatenate([y_test, pred], 0)
ptp = np.ptp(values)
min_value = np.min(values) - ptp * 0.1
max_value = np.max(values) + ptp * 0.1

plt.scatter(y_test, pred)
plt.plot([min_value, max_value], [min_value, max_value])
plt.xlim(min_value, max_value)
plt.ylim(min_value, max_value)
plt.xlabel('correct')
plt.ylabel('predict')
plt.title('Estimation of personal crimes using NN (objective variable: n
plt.show()
plt.savefig('res/23_personal_NN.png')
```


Estimation of personal crimes using NN (objective variable: no of crimes)
predict
correct
0
20
40
60
80
100
120
140
160

In [348]:

```
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import r2_score

data_test_all = crime_both_census.sample(n=100)
data_train_all = crime_both_census.loc[list(set(crime_both_census.index.

y_train = data_train_all['sum'].values.tolist()
data_train = data_train_all.drop("sum", 1)
x_train = data_train.values.tolist()

y_test = data_test_all['sum'].values.tolist()
data_test = data_test_all.drop("sum", 1)
x_test = data_test.values.tolist()

mlp = MLPRegressor(hidden_layer_sizes=(100,), max_iter=500) # https://nu
mlp.fit(x_train, y_train)
pred = mlp.predict(x_test)

%matplotlib notebook
values =  np.concatenate([y_test, pred], 0)
ptp = np.ptp(values)
min_value = np.min(values) - ptp * 0.1
max_value = np.max(values) + ptp * 0.1

plt.scatter(y_test, pred)
plt.plot([min_value, max_value], [min_value, max_value])
plt.xlim(min_value, max_value)
plt.ylim(min_value, max_value)
plt.xlabel('correct')
plt.ylabel('predict')
plt.title('Estimation of both crimes using NN (objective variable: no of
plt.show()
plt.savefig('res/24_both_NN.png')
```



Get total crime amount for each census

In [400]:

```
def groupby_taxiz_timep(_tmp):
    _out = pd.DataFrame(index=[], columns=['Taxi_Zone', 'Time_Period', '
    for i in _tmp['Taxi_Zone'].unique().tolist():
        for j in _tmp['Time_Period'].unique().tolist():
            __tmp = _tmp.query('Taxi_Zone == @i and Time_Period == @j')
            res = pd.Series([str(int(i)), j, sum(__tmp['total'].tolist()
            _out = pd.concat([_out, pd.DataFrame([res])])
    return _out

def groupby_crimes_timep(_tmp):
    _out = pd.DataFrame(index=[], columns=['Census_Tracts', 'Time_Period
    for i in _tmp['Census_Tracts'].unique().tolist():
        for j in _tmp['Time_Period'].unique().tolist():
            __tmp = _tmp.query('Census_Tracts == @i and Time_Period == @
            res = pd.Series([str(int(i)), j, sum(__tmp['total'].tolist()
            _out = pd.concat([_out, pd.DataFrame([res])])
    return _out

crime_property = pd.read_csv('1property_near_sorted_count.csv')
crime_property = crime_property[crime_property['Date'] != '(blank)']
crime_personal = pd.read_csv('1personal_near_sorted_count.csv')
crime_both = pd.concat([crime_property, crime_personal])

taxi = pd.read_csv('yellow_tripdata_2017_out.csv')

taxidata = taxi.groupby(['Taxi_Zone', 'Time_Period', 'Volume']).size().r
taxidata['total'] = taxidata['Volume'] * taxidata[0]
taxidata = groupby_taxiz_timep(taxidata)
taxidata.to_csv('res/taxi/taxi_timeperiod_dist.csv', index=False)

crime_property = crime_property.groupby(['Census_Tracts', 'Time_Period',
crime_property['total'] = crime_property['Volume'] * crime_property[0]
crime_property = groupby_crimes_timep(crime_property)
crime_property.to_csv('res/crime_dist/crime_property_timeperiod_dist.csv

crime_personal = crime_personal.groupby(['Census_Tracts', 'Time_Period',
crime_personal['total'] = crime_personal['Volume'] * crime_personal[0]
crime_personal = groupby_crimes_timep(crime_personal)
crime_personal.to_csv('res/crime_dist/crime_personal_timeperiod_dist.csv
```

```
crime_both = crime_both.groupby(['Census_Tracts', 'Time_Period', 'Volume
crime_both['total'] = crime_both['Volume'] * crime_both[0]
crime_both = groupby_crimes_timep(crime_both)
crime_both.to_csv('res/crime_dist/crime_both_timeperiod_dist.csv', index=
```

Some are output to taxi/crime_dist directory with time period data accoordingly visualise them

In [ ]:

```
def groupby_taxiz_total(_tmp):
    _out = pd.DataFrame(index=[], columns=['Taxi_Zone', 'total'])
    for i in _tmp['Taxi_Zone'].unique().tolist():
        __tmp = _tmp.query('Taxi_Zone == @i')
        res = pd.Series([str(int(i)), sum(__tmp['total'].tolist())], ind
        _out = pd.concat([_out, pd.DataFrame([res])])
    return _out

def groupby_crimes_total(_tmp):
    _out = pd.DataFrame(index=[], columns=['Census_Tracts', 'total'])
    for i in _tmp['Census_Tracts'].unique().tolist():
        __tmp = _tmp.query('Census_Tracts == @i')
        res = pd.Series([str(int(i)), sum(__tmp['total'].tolist())], ind
        _out = pd.concat([_out, pd.DataFrame([res])])
    return _out

taxidata = groupby_taxiz_total(taxidata).reset_index(drop=True)
crime_property = groupby_crimes_total(crime_property).reset_index(drop=T
crime_personal = groupby_crimes_total(crime_personal).reset_index(drop=T
crime_both = groupby_crimes_total(crime_both).reset_index(drop=True)
```

Taxi

In [440]:

```
taxidata = taxidata.astype({'Taxi_Zone': 'int', 'total': 'int'})
crime_property = crime_property.astype({'Census_Tracts': 'int'})
crime_personal = crime_personal.astype({'Census_Tracts': 'int'})
crime_both = crime_both.astype({'Census_Tracts': 'int'})

table_census_table = pd.read_csv('2010ct_jointaxi.csv')
table_census_table = pd.merge(table_census_table, taxidata, left_on='Tax
table_census_table_crime_property = pd.merge(table_census_table, crime_p
table_census_table_crime_personal = pd.merge(table_census_table, crime_p
table_census_table_crime_both = pd.merge(table_census_table, crime_both,
```

In [444]:

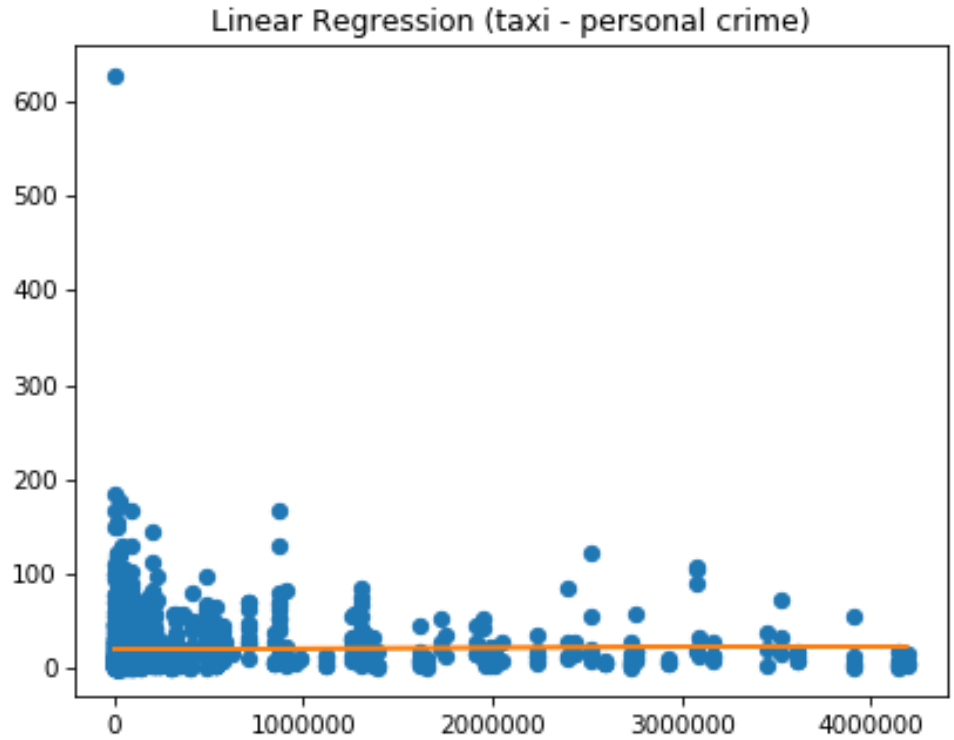
```
s1=pd.Series(table_census_table_crime_property['total_x'].tolist())
s2=pd.Series(table_census_table_crime_property['total_y'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = table_census_table_crime_property[['total_y']]
x = table_census_table_crime_property[['total_x']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (taxi - property crime)')
plt.show()
plt.savefig('res/25_taxi_crimeproperty.png')
```

```
Pearson correlation coefficient:  0.4847945002228223
```

In [445]:

```
s1=pd.Series(table_census_table_crime_personal['total_x'].tolist())
s2=pd.Series(table_census_table_crime_personal['total_y'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = table_census_table_crime_personal[['total_y']]
x = table_census_table_crime_personal[['total_x']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (taxi - personal crime)')
plt.show()
plt.savefig('res/26_taxi_crimepersonal.png')
```

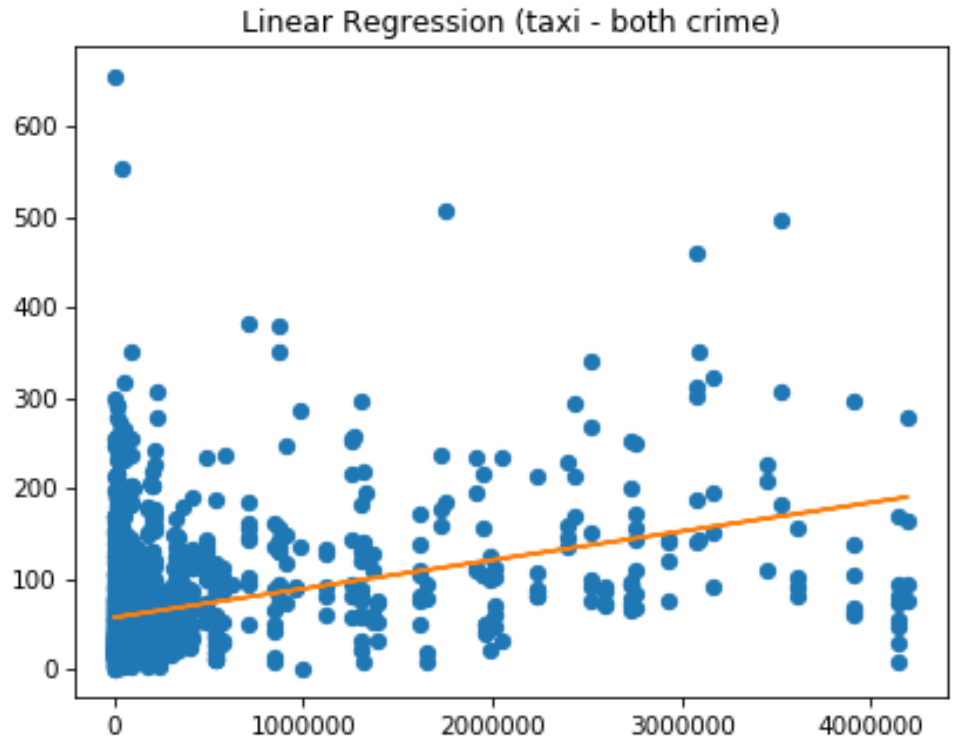
```
Pearson correlation coefficient:  0.01462012949131444
```



In [446]:

```
s1=pd.Series(table_census_table_crime_both['total_x'].tolist())
s2=pd.Series(table_census_table_crime_both['total_y'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = table_census_table_crime_both[['total_y']]
x = table_census_table_crime_both[['total_x']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (taxi - both crime)')
plt.show()
plt.savefig('res/27_taxi_crimeboth.png')
```

```
Pearson correlation coefficient:  0.35161446911771305
```



Past crimes

In [463]:

```
def unixAdd(_ds):
    return int(datetime.strptime(_ds.Date + "/" + _ds.Time, '%Y-%m-%d/%H

def cal_mean(_tmp):
    _out = pd.DataFrame(index=[], columns=['Census_Tracts', 'mean'])
    for i in _tmp['Census_Tracts'].unique().tolist():
        __tmp = _tmp.query('Census_Tracts == @i')
        res = pd.Series([str(int(i)), sum(__tmp['total'].tolist())/sum(_
        _out = pd.concat([_out, pd.DataFrame([res])])
    return _out

pastcrimes = pd.read_csv('3past_crimes.csv')
pastcrimes['unix'] = pastcrimes.apply(unixAdd,axis=1)

pastcrimes_property = pastcrimes.query('Type == "PropertyCrime"')
pastcrimes_personal = pastcrimes.query('Type == "PersonalCrime"')

per_2D = pd.DataFrame(pastcrimes_personal.groupby(['Census_Tracts', 'per
per_2D['total'] = per_2D['per_2D'] * per_2D[0]
per_2D = cal_mean(per_2D)
per_2D.to_csv('res/pastcrime/per_2D.csv', index=False)

per_7D = pd.DataFrame(pastcrimes_personal.groupby(['Census_Tracts', 'per
per_7D['total'] = per_7D['per_7D'] * per_7D[0]
per_7D = cal_mean(per_7D)
per_7D.to_csv('res/pastcrime/per_7D.csv', index=False)

per_30D = pd.DataFrame(pastcrimes_personal.groupby(['Census_Tracts', 'pe
per_30D['total'] = per_30D['per_30D'] * per_30D[0]
per_30D = cal_mean(per_30D)
per_30D.to_csv('res/pastcrime/per_30D.csv', index=False)

pro_2D = pd.DataFrame(pastcrimes_property.groupby(['Census_Tracts', 'pro
pro_2D['total'] = pro_2D['pro_2D'] * pro_2D[0]
pro_2D = cal_mean(pro_2D)
pro_2D.to_csv('res/pastcrime/pro_2D.csv', index=False)

pro_7D = pd.DataFrame(pastcrimes_property.groupby(['Census_Tracts', 'pro
pro_7D['total'] = pro_7D['pro_7D'] * pro_7D[0]
pro_7D = cal_mean(pro_7D)
```

```
pro_7D.to_csv('res/pastcrime/pro_7D.csv', index=False)

pro_30D = pd.DataFrame(pastcrimes_property.groupby(['Census_Tracts', 'pr
pro_30D['total'] = pro_30D['pro_30D'] * pro_30D[0]
pro_30D = cal_mean(pro_30D)
pro_30D.to_csv('res/pastcrime/pro_30D.csv', index=False)

both_2D = pd.DataFrame(pastcrimes.groupby(['Census_Tracts', 'both_2D']).
both_2D['total'] = both_2D['both_2D'] * both_2D[0]
both_2D = cal_mean(both_2D)
both_2D.to_csv('res/pastcrime/both_2D.csv', index=False)

both_7D = pd.DataFrame(pastcrimes.groupby(['Census_Tracts', 'both_7D']).
both_7D['total'] = both_7D['both_7D'] * both_7D[0]
both_7D = cal_mean(both_7D)
both_7D.to_csv('res/pastcrime/both_7D.csv', index=False)

both_30D = pd.DataFrame(pastcrimes.groupby(['Census_Tracts', 'both_30D']
both_30D['total'] = both_30D['both_30D'] * both_30D[0]
both_30D = cal_mean(both_30D)
both_30D.to_csv('res/pastcrime/both_30D.csv', index=False)

nbr_per_2D = pd.DataFrame(pastcrimes_personal.groupby(['Census_Tracts',
nbr_per_2D['total'] = nbr_per_2D['nbr_per_2D'] * nbr_per_2D[0]
nbr_per_2D = cal_mean(nbr_per_2D)
nbr_per_2D.to_csv('res/pastcrime/nbr_per_2D.csv', index=False)

nbr_per_7D = pd.DataFrame(pastcrimes_personal.groupby(['Census_Tracts',
nbr_per_7D['total'] = nbr_per_7D['nbr_per_7D'] * nbr_per_7D[0]
nbr_per_7D = cal_mean(nbr_per_7D)
nbr_per_7D.to_csv('res/pastcrime/nbr_per_7D.csv', index=False)

nbr_per_30D = pd.DataFrame(pastcrimes_personal.groupby(['Census_Tracts',
nbr_per_30D['total'] = nbr_per_30D['nbr_per_30D'] * nbr_per_30D[0]
nbr_per_30D = cal_mean(nbr_per_30D)
nbr_per_30D.to_csv('res/pastcrime/nbr_per_30D.csv', index=False)

nbr_pro_2D = pd.DataFrame(pastcrimes_property.groupby(['Census_Tracts',
nbr_pro_2D['total'] = nbr_pro_2D['nbr_pro_2D'] * nbr_pro_2D[0]
nbr_pro_2D = cal_mean(nbr_pro_2D)
nbr_pro_2D.to_csv('res/pastcrime/nbr_pro_2D.csv', index=False)

nbr_pro_7D = pd.DataFrame(pastcrimes_property.groupby(['Census_Tracts',
```

```
nbr_pro_7D['total'] = nbr_pro_7D['nbr_pro_7D'] * nbr_pro_7D[0]
nbr_pro_7D = cal_mean(nbr_pro_7D)
nbr_pro_7D.to_csv('res/pastcrime/nbr_pro_7D.csv', index=False)

nbr_pro_30D = pd.DataFrame(pastcrimes_property.groupby(['Census_Tracts',
nbr_pro_30D['total'] = nbr_pro_30D['nbr_pro_30D'] * nbr_pro_30D[0]
nbr_pro_30D = cal_mean(nbr_pro_30D)
nbr_pro_30D.to_csv('res/pastcrime/nbr_pro_30D.csv', index=False)

nbr_both_2D = pd.DataFrame(pastcrimes.groupby(['Census_Tracts', 'nbr_bot
nbr_both_2D['total'] = nbr_both_2D['nbr_both_2D'] * nbr_both_2D[0]
nbr_both_2D = cal_mean(nbr_both_2D)
nbr_both_2D.to_csv('res/pastcrime/nbr_both_2D.csv', index=False)

nbr_both_7D = pd.DataFrame(pastcrimes.groupby(['Census_Tracts', 'nbr_bot
nbr_both_7D['total'] = nbr_both_7D['nbr_both_7D'] * nbr_both_7D[0]
nbr_both_7D = cal_mean(nbr_both_7D)
nbr_both_7D.to_csv('res/pastcrime/nbr_both_7D.csv', index=False)

nbr_both_30D = pd.DataFrame(pastcrimes.groupby(['Census_Tracts', 'nbr_bo
nbr_both_30D['total'] = nbr_both_30D['nbr_both_30D'] * nbr_both_30D[0]
nbr_both_30D = cal_mean(nbr_both_30D)
nbr_both_30D.to_csv('res/pastcrime/nbr_both_30D.csv', index=False)
```

The following outputs indicates the temporal continuity (on avrage by all the crimes) of crimes for each census unit (the higher the more dangerous), accordinly visualise them.

- per_2D.csv (the contuity of personal crimes happend in the same census in the past 2days)
- per_7D.csv
- per_30D.csv
- pro_2D.csv (the contuity of property crimes happend in the same census in past 2days)
- pro_7D.csv
- pro_30D.csv
- both_2D.csv (the contuity of boths (mixed) crimes happend in the same census in past 2days)
- both_7D.csv
- both_30D.csv
- nbr_per_2D.csv (the contuity of personal crimes happend in the same neighbor census in the past 2days)
- nbr_per_7D.csv
- nbr_per_30D.csv
- nbr_pro_2D.csv (the contuity of property crimes happend in the same neighbor census in past 2days)

- nbr_pro_7D.csv
- nbr_pro_30D.csv
- nbr_both_2D.csv (the contuity of boths (mixed) crimes happend in the neighbor census in past 2days)
- nbr_both_7D.csv
- nbr_both_30D.csv

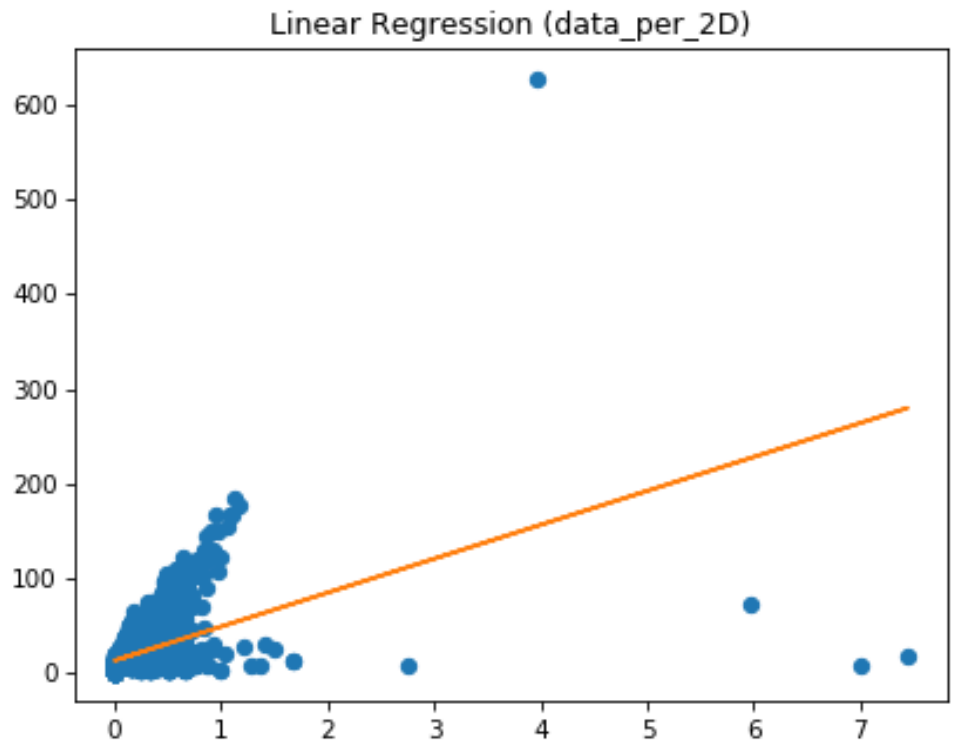
In [464]:

```
crime_property = crime_property.astype({'Census_Tracts': 'int'})
crime_personal = crime_personal.astype({'Census_Tracts': 'int'})
crime_both = crime_both.astype({'Census_Tracts': 'int'})

data_per_2D = pd.merge(per_2D.astype({'Census_Tracts': 'int'}), crime_pe
data_per_7D = pd.merge(per_7D.astype({'Census_Tracts': 'int'}), crime_pe
data_per_30D = pd.merge(per_30D.astype({'Census_Tracts': 'int'}), crime_

data_pro_2D = pd.merge(pro_2D.astype({'Census_Tracts': 'int'}), crime_pr
data_pro_7D = pd.merge(pro_7D.astype({'Census_Tracts': 'int'}), crime_pr
data_pro_30D = pd.merge(pro_30D.astype({'Census_Tracts': 'int'}), crime_

data_both_2D = pd.merge(both_2D.astype({'Census_Tracts': 'int'}), crime_
data_both_7D = pd.merge(both_7D.astype({'Census_Tracts': 'int'}), crime_
data_both_30D = pd.merge(both_30D.astype({'Census_Tracts': 'int'}), crim

data_nbr_per_2D = pd.merge(nbr_per_2D.astype({'Census_Tracts': 'int'}),
data_nbr_per_7D = pd.merge(nbr_per_7D.astype({'Census_Tracts': 'int'}),
data_nbr_per_30D = pd.merge(nbr_per_30D.astype({'Census_Tracts': 'int'})

data_nbr_pro_2D = pd.merge(nbr_pro_2D.astype({'Census_Tracts': 'int'}),
data_nbr_pro_7D = pd.merge(nbr_pro_7D.astype({'Census_Tracts': 'int'}),
data_nbr_pro_30D = pd.merge(nbr_pro_30D.astype({'Census_Tracts': 'int'})

data_nbr_both_2D = pd.merge(nbr_both_2D.astype({'Census_Tracts': 'int'})
data_nbr_both_7D = pd.merge(nbr_both_7D.astype({'Census_Tracts': 'int'})
data_nbr_both_30D = pd.merge(nbr_both_30D.astype({'Census_Tracts': 'int'
```

In [472]:

```
s1=pd.Series(data_per_2D['total'].tolist())
s2=pd.Series(data_per_2D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_per_2D[['total']]
x = data_per_2D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_per_2D)')
plt.show()
plt.savefig('res/28_pastcrimes_data_per_2D.png')
```

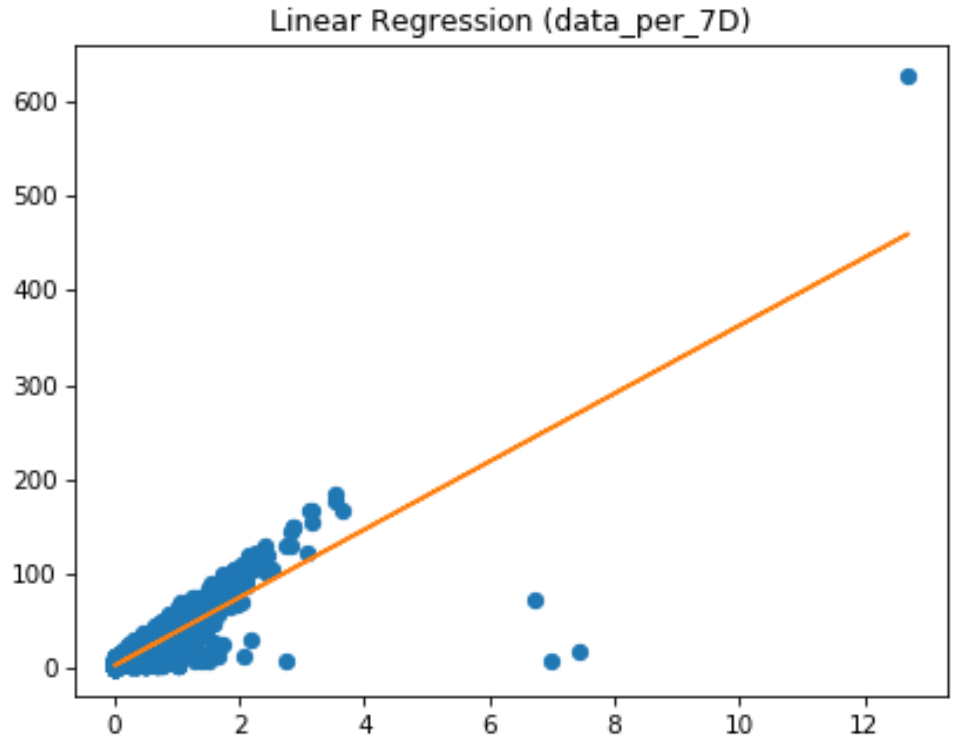
```
Pearson correlation coefficient:  0.47684117718704055
```



In [473]:

```
s1=pd.Series(data_per_7D['total'].tolist())
s2=pd.Series(data_per_7D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_per_7D[['total']]
x = data_per_7D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_per_7D)')
plt.show()
plt.savefig('res/29_pastcrimes_data_per_7D.png')
```

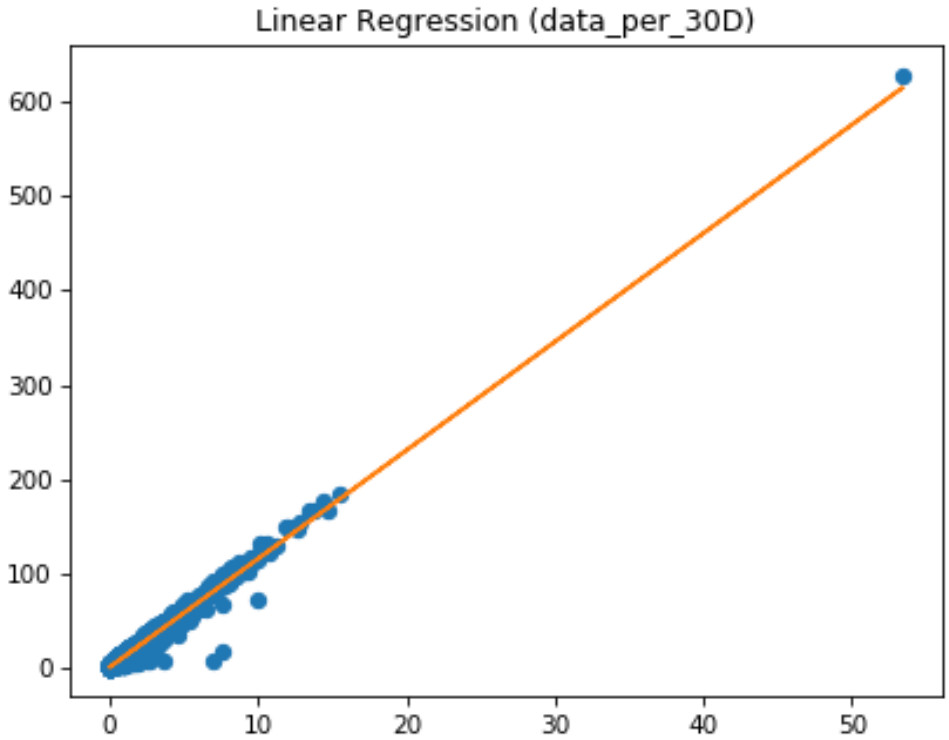
```
Pearson correlation coefficient:  0.8497672400592429
```



In [474]:

```
s1=pd.Series(data_per_30D['total'].tolist())
s2=pd.Series(data_per_30D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_per_30D[['total']]
x = data_per_30D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_per_30D)')
plt.show()
plt.savefig('res/30_pastcrimes_data_per_30D.png')
```

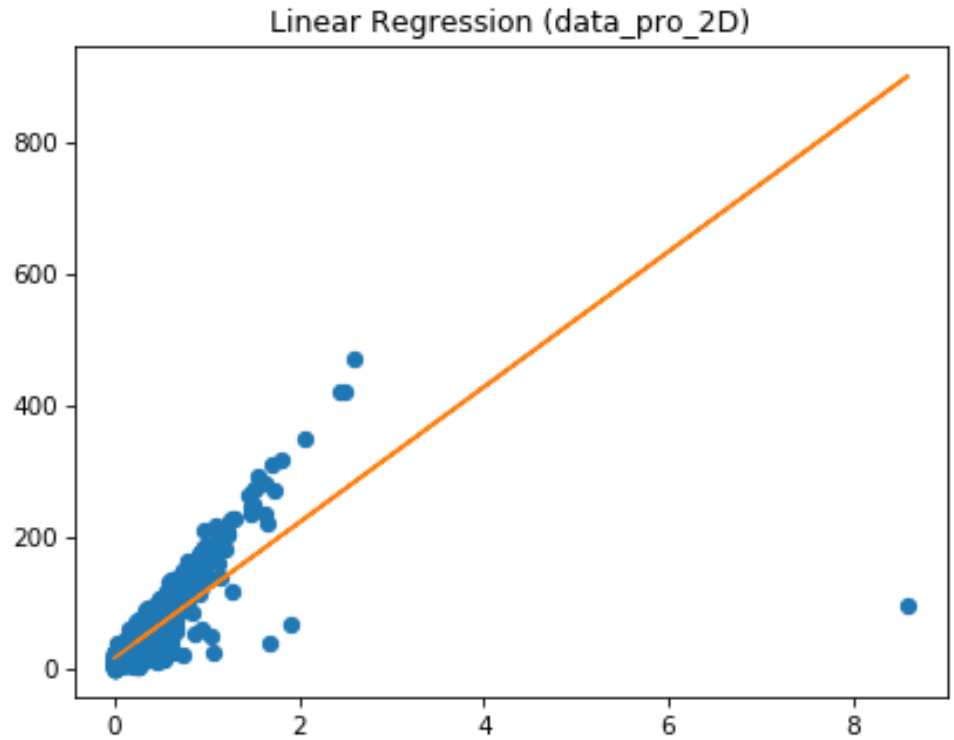
```
Pearson correlation coefficient:  0.9826170708118905
```



In [475]:

```
s1=pd.Series(data_pro_2D['total'].tolist())
s2=pd.Series(data_pro_2D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_pro_2D[['total']]
x = data_pro_2D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_pro_2D)')
plt.show()
plt.savefig('res/31_pastcrimes_data_pro_2D.png')
```

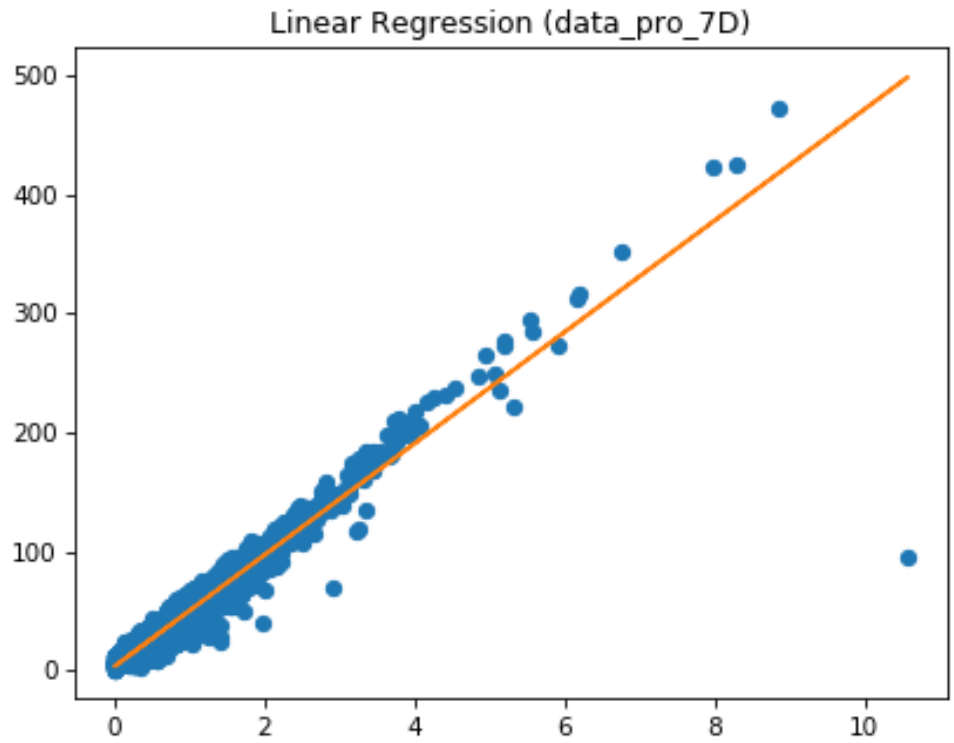
```
Pearson correlation coefficient:  0.7755768978142796
```



In [476]:

```
s1=pd.Series(data_pro_7D['total'].tolist())
s2=pd.Series(data_pro_7D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_pro_7D[['total']]
x = data_pro_7D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_pro_7D)')
plt.show()
plt.savefig('res/32_pastcrimes_data_pro_7D.png')
```

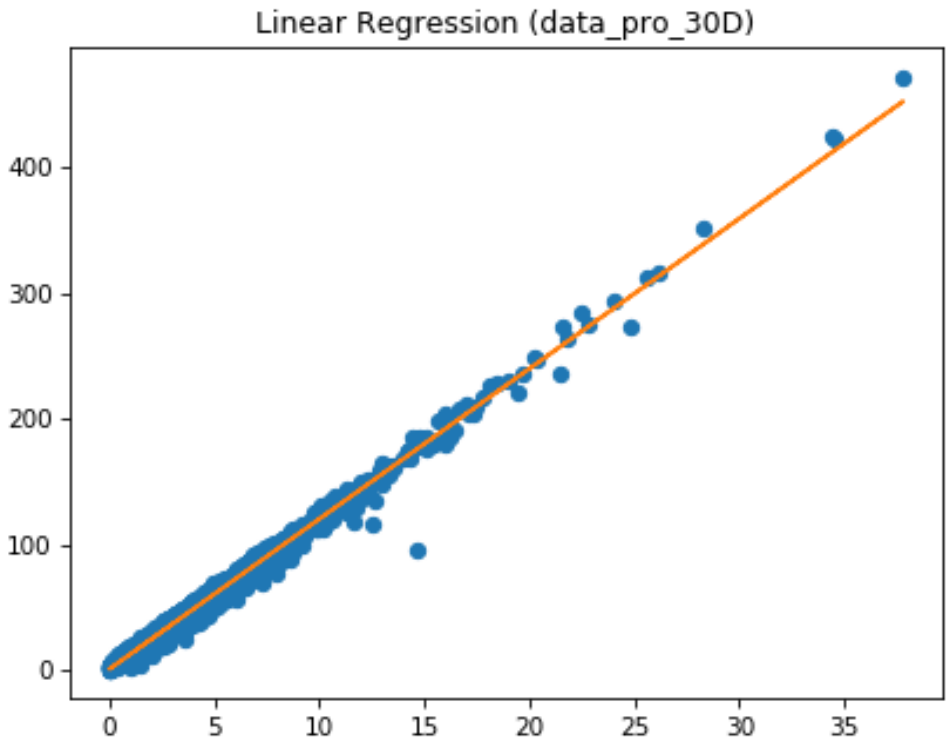
```
Pearson correlation coefficient:  0.9578123080606001
```



In [477]:

```
s1=pd.Series(data_pro_30D['total'].tolist())
s2=pd.Series(data_pro_30D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_pro_30D[['total']]
x = data_pro_30D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_pro_30D)')
plt.show()
plt.savefig('res/33_pastcrimes_data_pro_30D.png')
```

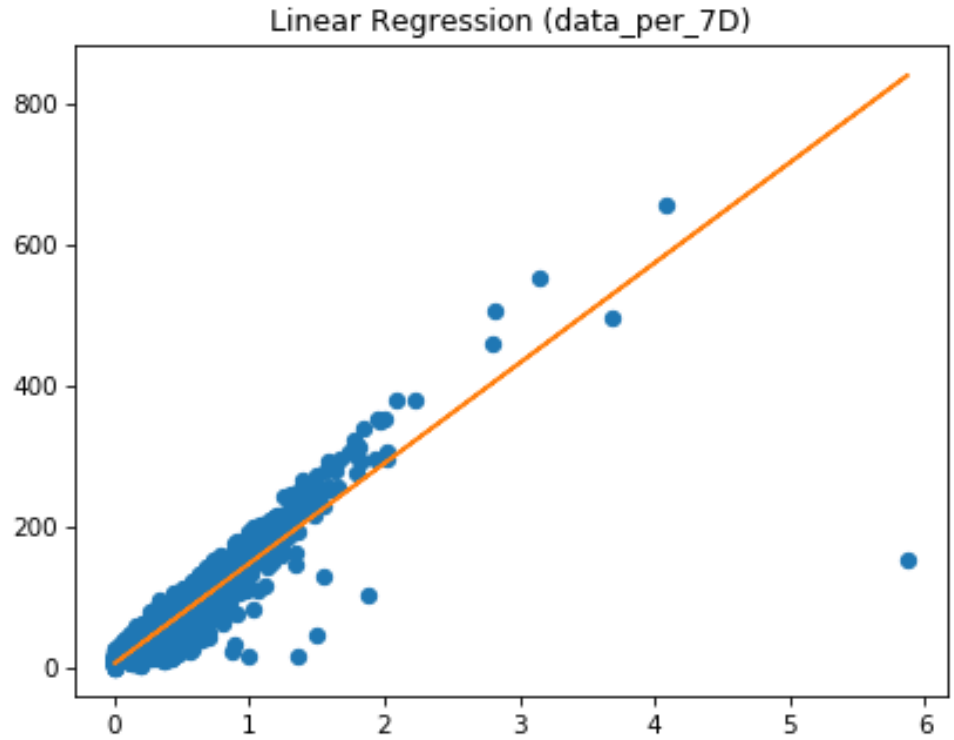
```
Pearson correlation coefficient:  0.9943658223353585
```



In [478]:

```
s1=pd.Series(data_both_2D['total'].tolist())
s2=pd.Series(data_both_2D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_both_2D[['total']]
x = data_both_2D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_per_7D)')
plt.show()
plt.savefig('res/34_pastcrimes_data_both_2D.png')
```

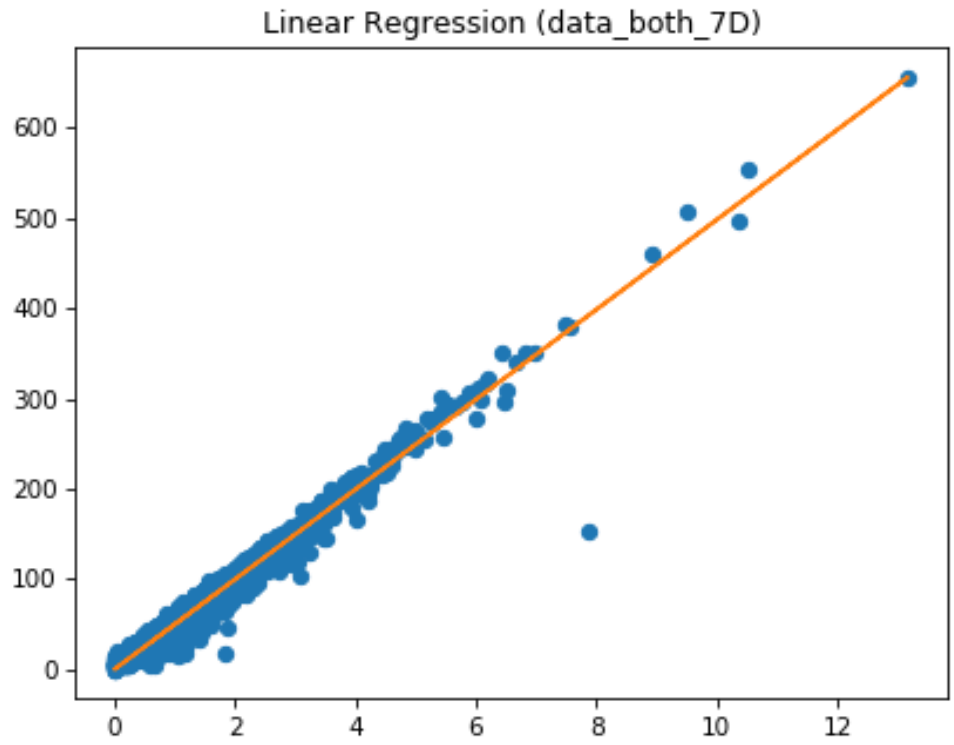
```
Pearson correlation coefficient:  0.9098709271004599
```



In [482]:

```
s1=pd.Series(data_both_7D['total'].tolist())
s2=pd.Series(data_both_7D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_both_7D[['total']]
x = data_both_7D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_both_7D)')
plt.show()
plt.savefig('res/36_pastcrimes_data_both_7D.png')
```

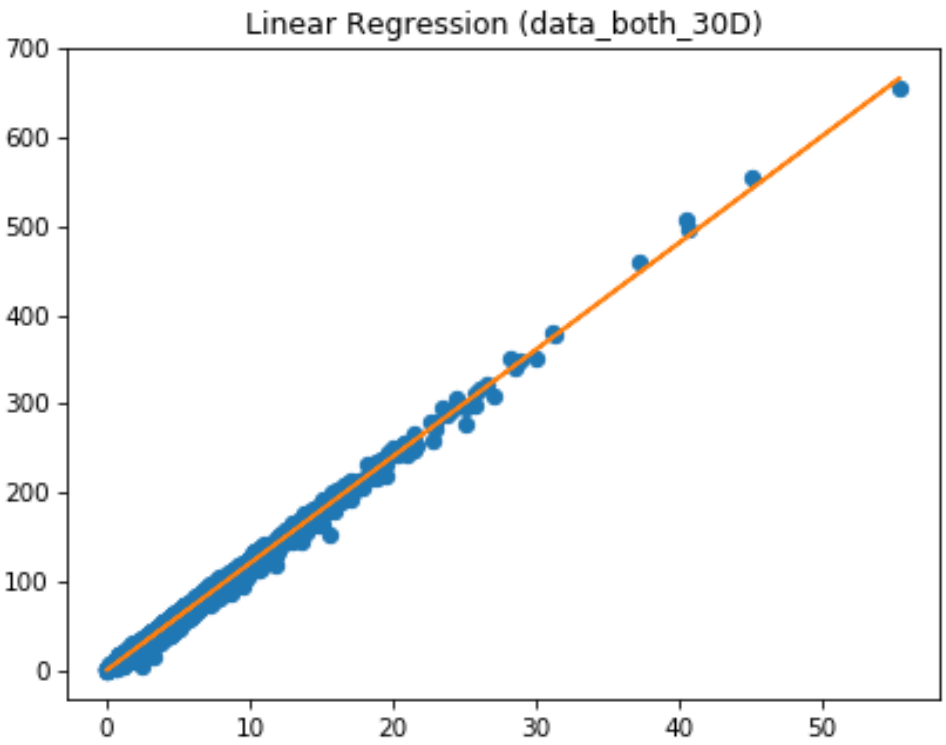
```
Pearson correlation coefficient:  0.9849678661670173
```



In [483]:

```
s1=pd.Series(data_both_30D['total'].tolist())
s2=pd.Series(data_both_30D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_both_30D[['total']]
x = data_both_30D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_both_30D)')
plt.show()
plt.savefig('res/37_pastcrimes_data_both_30D.png')
```

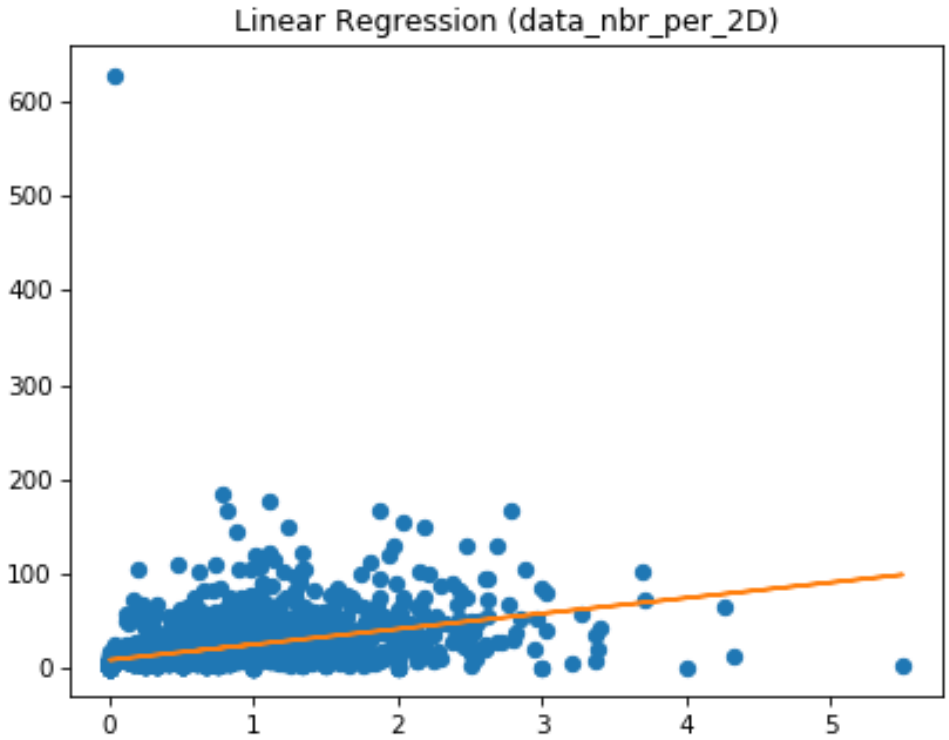
```
Pearson correlation coefficient:  0.9969959932952183
```



In [484]:

```
s1=pd.Series(data_nbr_per_2D['total'].tolist())
s2=pd.Series(data_nbr_per_2D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_nbr_per_2D[['total']]
x = data_nbr_per_2D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_nbr_per_2D)')
plt.show()
plt.savefig('res/38_pastcrimes_data_nbr_per_2D.png')
```

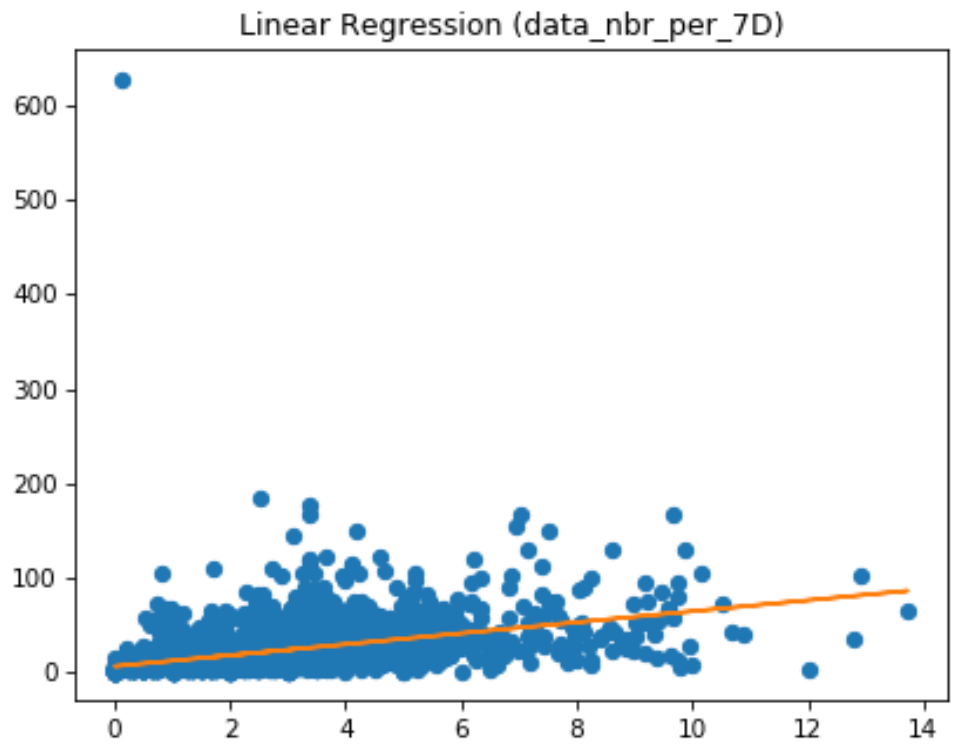
```
Pearson correlation coefficient:  0.40931608581189427
```



In [485]:

```
s1=pd.Series(data_nbr_per_7D['total'].tolist())
s2=pd.Series(data_nbr_per_7D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_nbr_per_7D[['total']]
x = data_nbr_per_7D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_nbr_per_7D)')
plt.show()
plt.savefig('res/39_pastcrimes_data_nbr_per_7D.png')
```

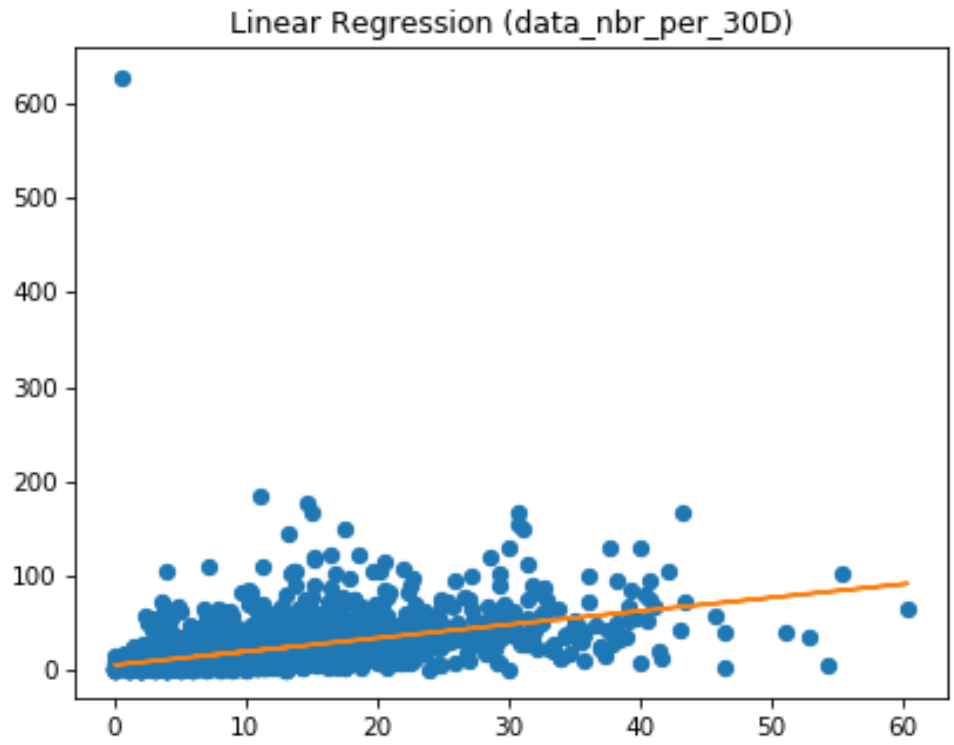
```
Pearson correlation coefficient:  0.4551175804121565
```



In [486]:

```
s1=pd.Series(data_nbr_per_30D['total'].tolist())
s2=pd.Series(data_nbr_per_30D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_nbr_per_30D[['total']]
x = data_nbr_per_30D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_nbr_per_30D)')
plt.show()
plt.savefig('res/40_pastcrimes_data_nbr_per_30D.png')
```

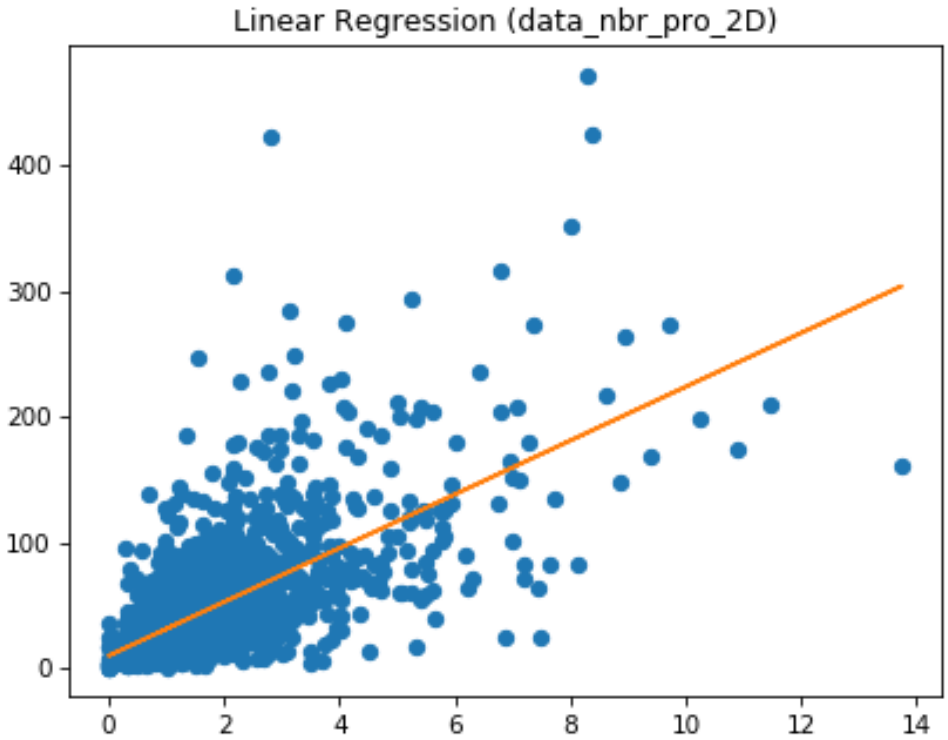
```
Pearson correlation coefficient:  0.46792548635819375
```



In [487]:

```
s1=pd.Series(data_nbr_pro_2D['total'].tolist())
s2=pd.Series(data_nbr_pro_2D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_nbr_pro_2D[['total']]
x = data_nbr_pro_2D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_nbr_pro_2D)')
plt.show()
plt.savefig('res/41_pastcrimes_data_nbr_pro_2D.png')
```

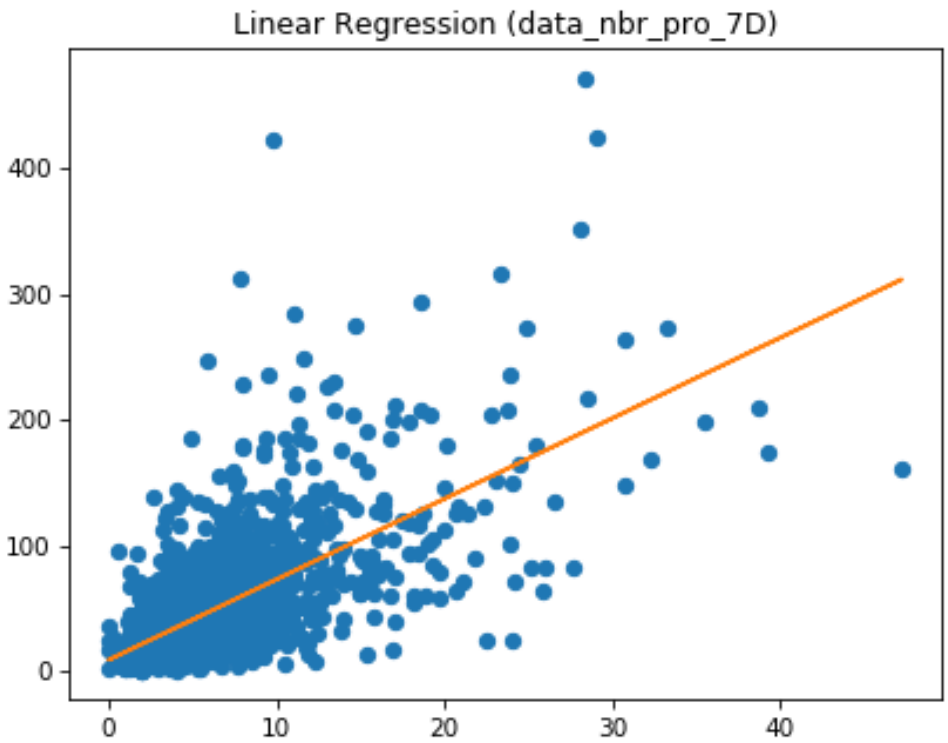
```
Pearson correlation coefficient:  0.6601460112299974
```



In [488]:

```
s1=pd.Series(data_nbr_pro_7D['total'].tolist())
s2=pd.Series(data_nbr_pro_7D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_nbr_pro_7D[['total']]
x = data_nbr_pro_7D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_nbr_pro_7D)')
plt.show()
plt.savefig('res/42_pastcrimes_data_nbr_pro_7D.png')
```

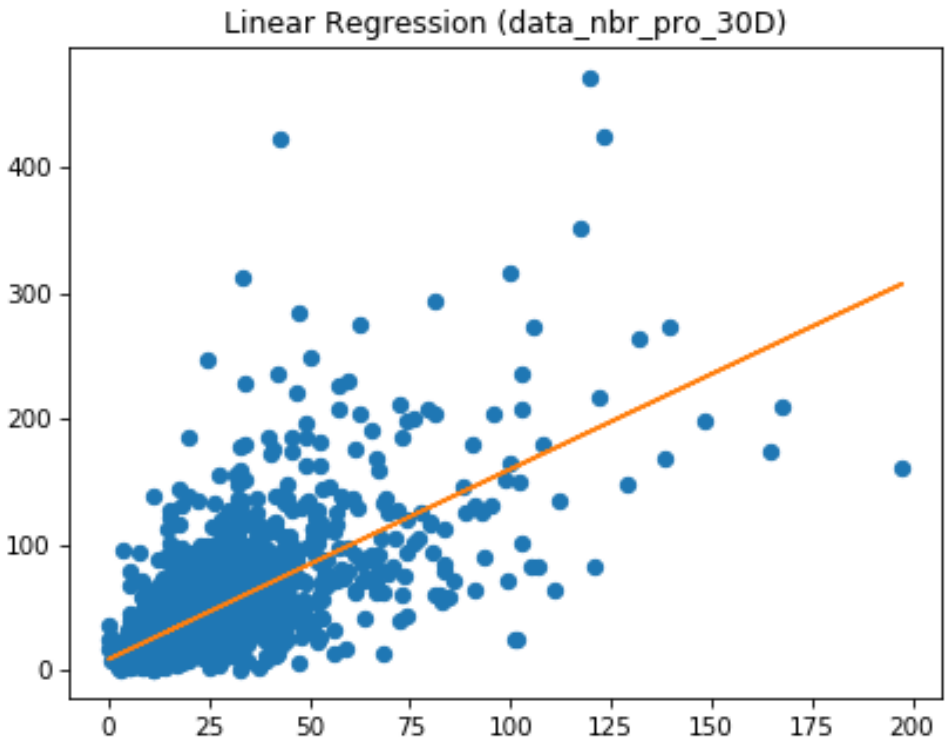
```
Pearson correlation coefficient:  0.6707554127067068
```



In [489]:

```
s1=pd.Series(data_nbr_pro_30D['total'].tolist())
s2=pd.Series(data_nbr_pro_30D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_nbr_pro_30D[['total']]
x = data_nbr_pro_30D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_nbr_pro_30D)')
plt.show()
plt.savefig('res/43_pastcrimes_data_nbr_pro_30D.png')
```

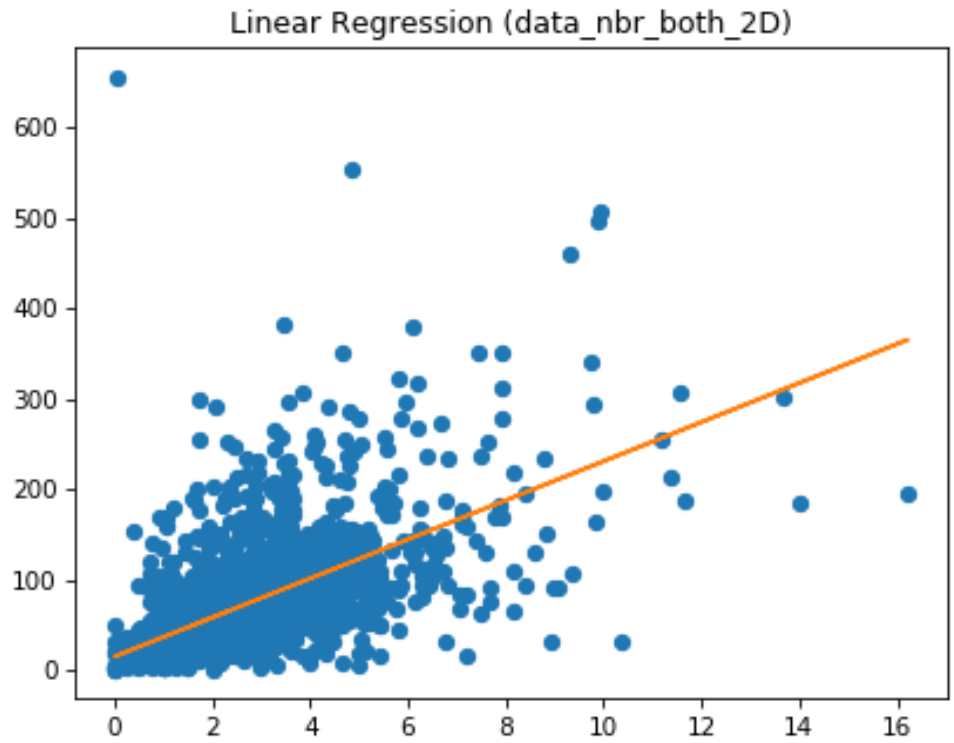
```
Pearson correlation coefficient:  0.6746757005678838
```



In [490]:

```
s1=pd.Series(data_nbr_both_2D['total'].tolist())
s2=pd.Series(data_nbr_both_2D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_nbr_both_2D[['total']]
x = data_nbr_both_2D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_nbr_both_2D)')
plt.show()
plt.savefig('res/44_pastcrimes_data_nbr_both_2D.png')
```

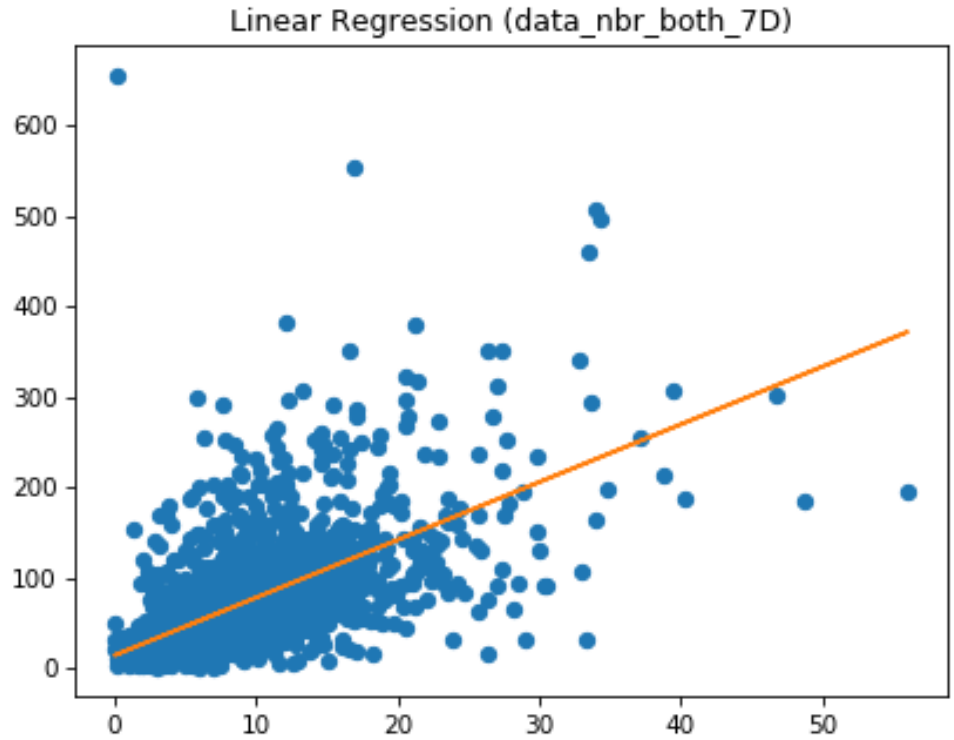
```
Pearson correlation coefficient:  0.6209140129407131
```



In [491]:

```
s1=pd.Series(data_nbr_both_7D['total'].tolist())
s2=pd.Series(data_nbr_both_7D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_nbr_both_7D[['total']]
x = data_nbr_both_7D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_nbr_both_7D)')
plt.show()
plt.savefig('res/45_pastcrimes_data_nbr_both_7D.png')
```

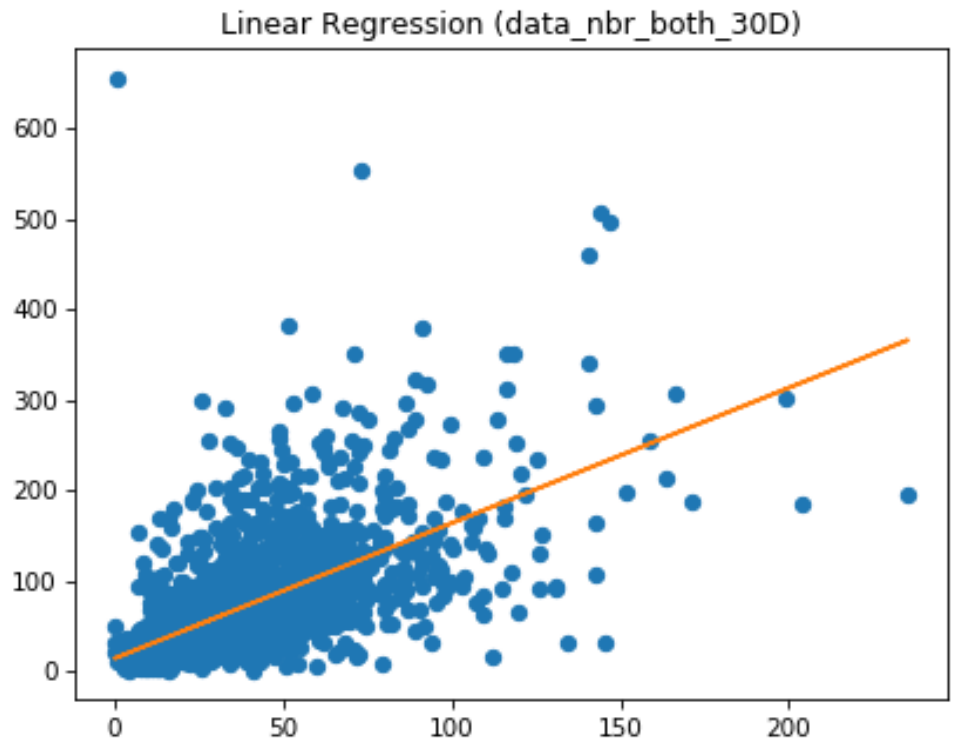
```
Pearson correlation coefficient:  0.6268534872678536
```



In [492]:

```
s1=pd.Series(data_nbr_both_30D['total'].tolist())
s2=pd.Series(data_nbr_both_30D['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_nbr_both_30D[['total']]
x = data_nbr_both_30D[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_nbr_both_30D)')
plt.show()
plt.savefig('res/46_pastcrimes_data_nbr_both_30D.png')
```

```
Pearson correlation coefficient:  0.6292674097365941
```



Frisk

In [497]:

```
def unixAdd_frisk(_ds):
    return int(datetime.strptime(_ds.Date + "/" + _ds.Time, '%m/%d/%Y/%H

def cal_mean_frisk(_tmp):
    _out = pd.DataFrame(index=[], columns=['Census_Tracts', 'mean'])
    for i in _tmp['boro_ct2010'].unique().tolist():
        __tmp = _tmp.query('boro_ct2010 == @i')
        res = pd.Series([str(int(i)), sum(__tmp['total'].tolist())/sum(_
        _out = pd.concat([_out, pd.DataFrame([res])])
    return _out

pastfrisk = pd.read_csv('crimes_pastfrisk_counts.csv')
pastfrisk['unix'] = pastfrisk.apply(unixAdd_frisk,axis=1)

pastfrisk_property = pastfrisk.query('Type == "PropertyCrime"')
pastfrisk_personal = pastfrisk.query('Type == "PersonalCrime"')

ct_property_frisk = pd.DataFrame(pastfrisk_property.groupby(['boro_ct201
ct_property_frisk['total'] = ct_property_frisk['count_ct'] * ct_property
ct_property_frisk = cal_mean_frisk(ct_property_frisk)
ct_property_frisk.to_csv('res/pastfrisk/ct_property_frisk.csv', index=Fa

nbr_property_frisk = pd.DataFrame(pastfrisk_property.groupby(['boro_ct20
nbr_property_frisk['total'] = nbr_property_frisk['count_neighbours'] * nb
nbr_property_frisk = cal_mean_frisk(nbr_property_frisk)
nbr_property_frisk.to_csv('res/pastfrisk/nbr_property_frisk.csv', index=

# both_property_frisk = pd.DataFrame(pastfrisk_property.groupby(['boro_c
# both_property_frisk['total'] = both_property_frisk['count_total'] * bo
# cal_mean_frisk(both_property_frisk).to_csv('res/pastfrisk/both_propert

ct_personal_frisk = pd.DataFrame(pastfrisk_personal.groupby(['boro_ct201
ct_personal_frisk['total'] = ct_personal_frisk['count_ct'] * ct_personal
ct_personal_frisk = cal_mean_frisk(ct_personal_frisk)
ct_personal_frisk.to_csv('res/pastfrisk/ct_personal_frisk.csv', index=Fa

nbr_personal_frisk = pd.DataFrame(pastfrisk_personal.groupby(['boro_ct20
nbr_personal_frisk['total'] = nbr_personal_frisk['count_neighbours'] * nb
nbr_personal_frisk = cal_mean_frisk(nbr_personal_frisk)
nbr_personal_frisk.to_csv('res/pastfrisk/nbr_personal_frisk.csv', index=
```

```
# both_personal_frisk = pd.DataFrame(pastfrisk_personal.groupby(['boro_c
# both_personal_frisk['total'] = both_personal_frisk['count_total'] * bo
# cal_mean_frisk(both_personal_frisk).to_csv('res/pastfrisk/both_persona

ct_both_frisk = pd.DataFrame(pastfrisk.groupby(['boro_ct2010', 'count_ct
ct_both_frisk['total'] = ct_both_frisk['count_ct'] * ct_both_frisk[0]
ct_both_frisk = cal_mean_frisk(ct_both_frisk)
ct_both_frisk.to_csv('res/pastfrisk/ct_both_frisk.csv', index=False)

nbr_both_frisk = pd.DataFrame(pastfrisk.groupby(['boro_ct2010', 'count_n
nbr_both_frisk['total'] = nbr_both_frisk['count_neighbours'] * nbr_both_
nbr_both_frisk = cal_mean_frisk(nbr_both_frisk)
nbr_both_frisk.to_csv('res/pastfrisk/nbr_both_frisk.csv', index=False)

# both_both_frisk = pd.DataFrame(pastfrisk.groupby(['boro_ct2010', 'coun
# both_both_frisk['total'] = both_both_frisk['count_total'] * both_both_
# cal_mean_frisk(both_both_frisk).to_csv('res/pastfrisk/both_both_frisk.
```

The following outputs indicates the temporal effect of police exsistence (on avrage by all the crimes) of crimes for each census unit (the higher the more secured), accordinly visualise them.

- ct_property_frisk.csv (frisk observed in a certain census in the past 6hrs regarding property crimes)
- nbr_property_frisk.csv (frisk observed in neighboring census in the past 6hrs regarding property crimes)
- ct_personal_frisk.csv (...)
- nbr_personal_frisk.csv
- ct_both_frisk.csv
- nbr_both_frisk.csv

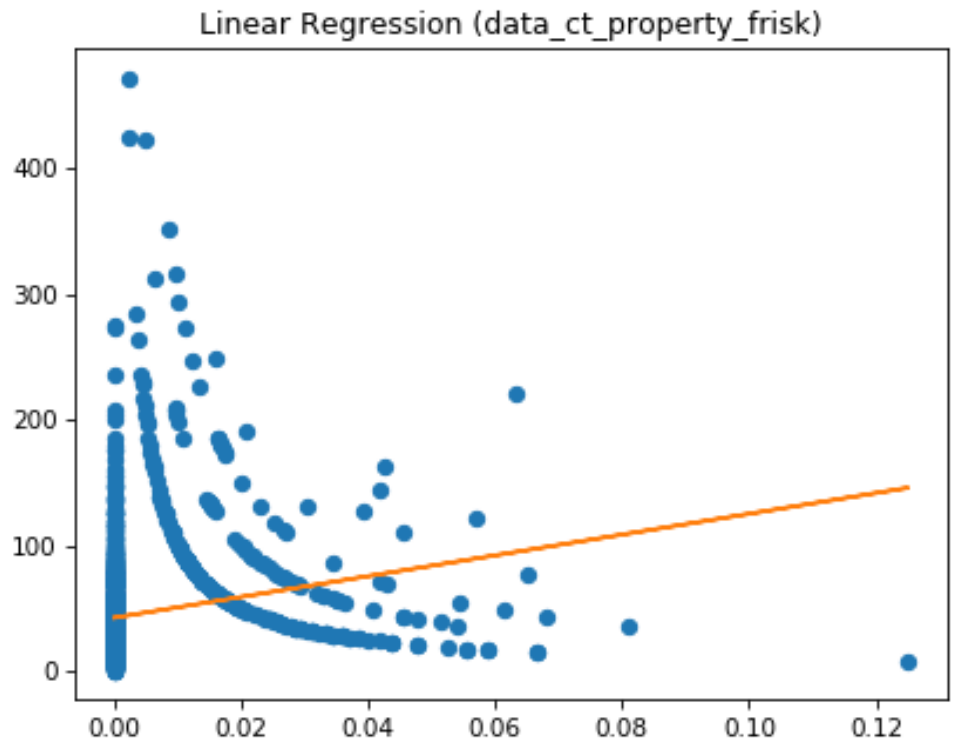
In [504]:

```
data_ct_property_frisk = pd.merge(ct_property_frisk.astype({'Census_Tracts
data_nbr_property_frisk= pd.merge(nbr_property_frisk.astype({'Census_Tract
data_ct_personal_frisk = pd.merge(ct_personal_frisk.astype({'Census_Tracts
data_nbr_personal_frisk = pd.merge(nbr_personal_frisk.astype({'Census_Trac
data_ct_both_frisk = pd.merge(ct_both_frisk.astype({'Census_Tracts': 'int'
data_nbr_both_frisk = pd.merge(nbr_both_frisk.astype({'Census_Tracts': 'in
```

In [507]:

```
s1=pd.Series(data_ct_property_frisk['total'].tolist())
s2=pd.Series(data_ct_property_frisk['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_ct_property_frisk[['total']]
x = data_ct_property_frisk[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_ct_property_frisk)')
plt.show()
plt.savefig('res/47_pastfrisk_data_ct_property_frisk.png')
```

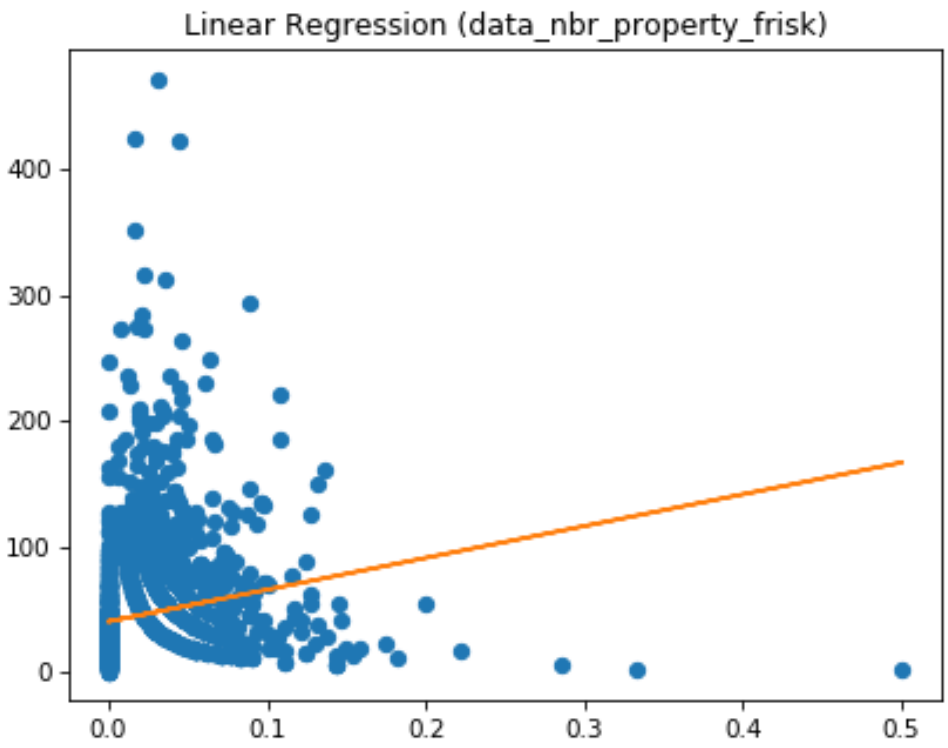
```
Pearson correlation coefficient:  0.18107630417359685
```



In [508]:

```
s1=pd.Series(data_nbr_property_frisk['total'].tolist())
s2=pd.Series(data_nbr_property_frisk['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_nbr_property_frisk[['total']]
x = data_nbr_property_frisk[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_nbr_property_frisk)')
plt.show()
plt.savefig('res/48_pastcrimes_data_nbr_property_frisk.png')
```

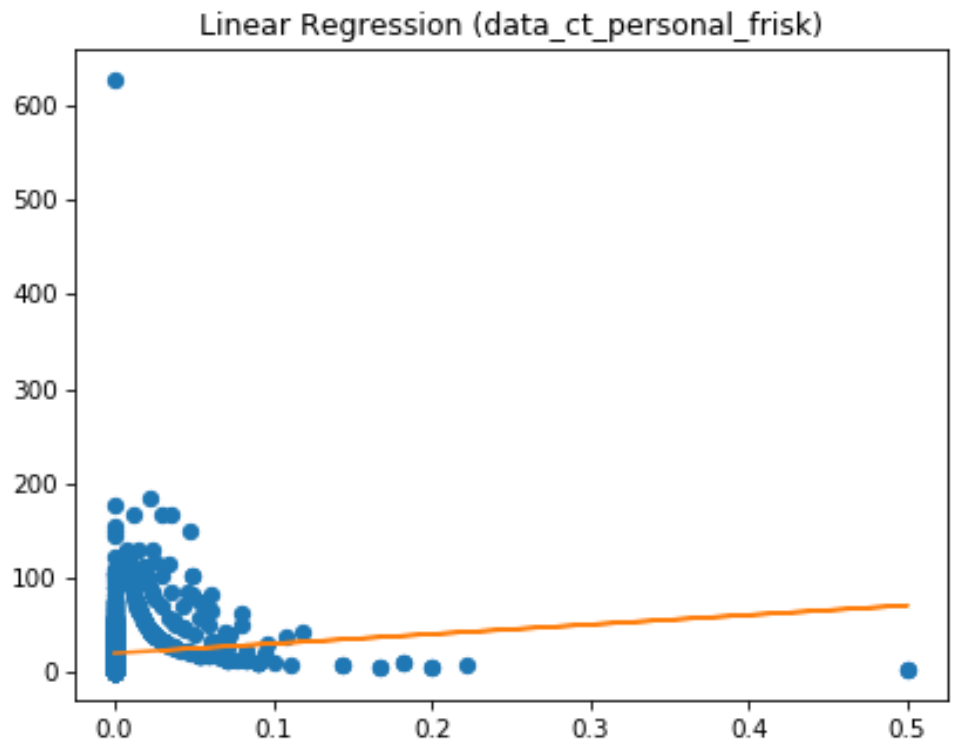
```
Pearson correlation coefficient:  0.1854832380745206
```



In [509]:

```
s1=pd.Series(data_ct_personal_frisk['total'].tolist())
s2=pd.Series(data_ct_personal_frisk['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_ct_personal_frisk[['total']]
x = data_ct_personal_frisk[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_ct_personal_frisk)')
plt.show()
plt.savefig('res/49_pastcrimes_data_ct_personal_frisk.png')
```

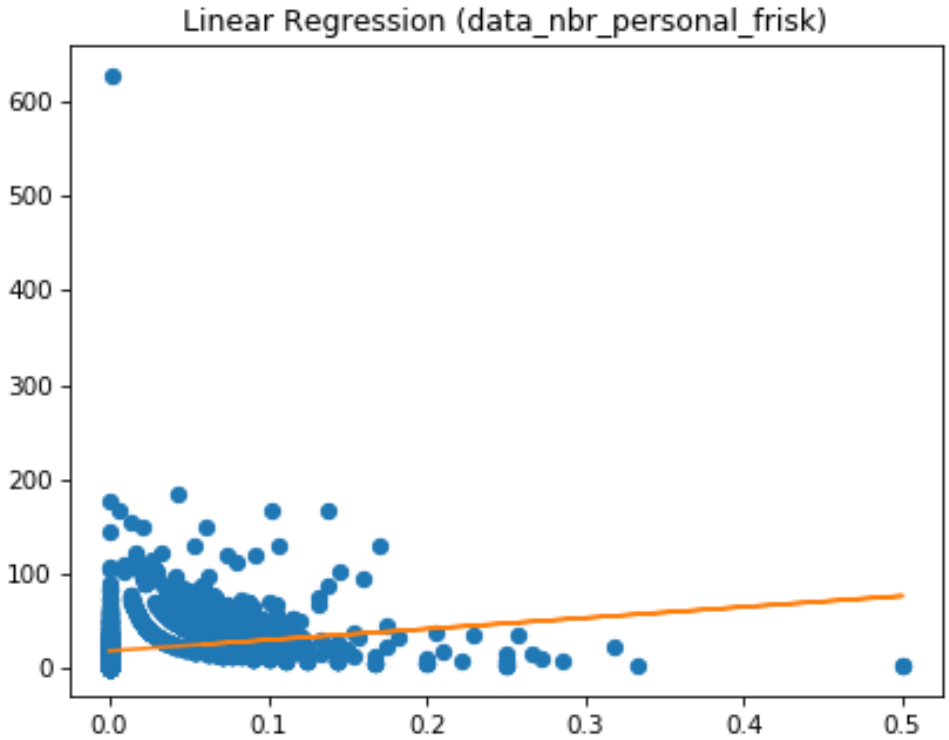
```
Pearson correlation coefficient:  0.09033925326790848
```



In [510]:

```
s1=pd.Series(data_nbr_personal_frisk['total'].tolist())
s2=pd.Series(data_nbr_personal_frisk['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_nbr_personal_frisk[['total']]
x = data_nbr_personal_frisk[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_nbr_personal_frisk)')
plt.show()
plt.savefig('res/50_pastcrimes_data_nbr_personal_frisk.png')
```

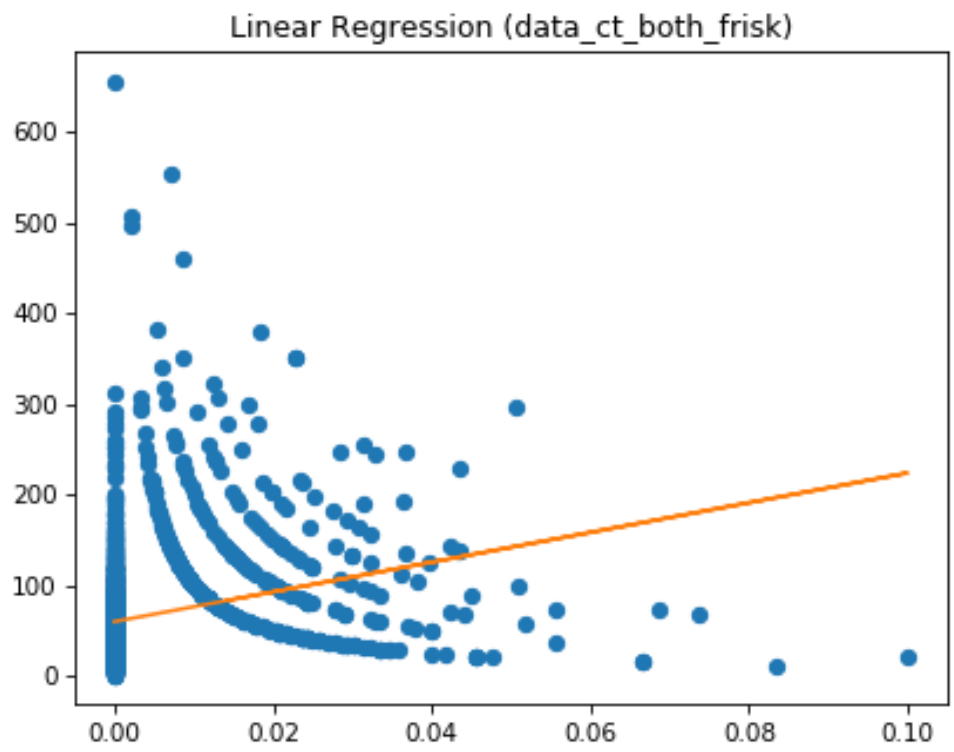
```
Pearson correlation coefficient:  0.18851312082417707
```



In [511]:

```
s1=pd.Series(data_ct_both_frisk['total'].tolist())
s2=pd.Series(data_ct_both_frisk['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_ct_both_frisk[['total']]
x = data_ct_both_frisk[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_ct_both_frisk)')
plt.show()
plt.savefig('res/51_pastcrimes_data_ct_both_frisk.png')
```

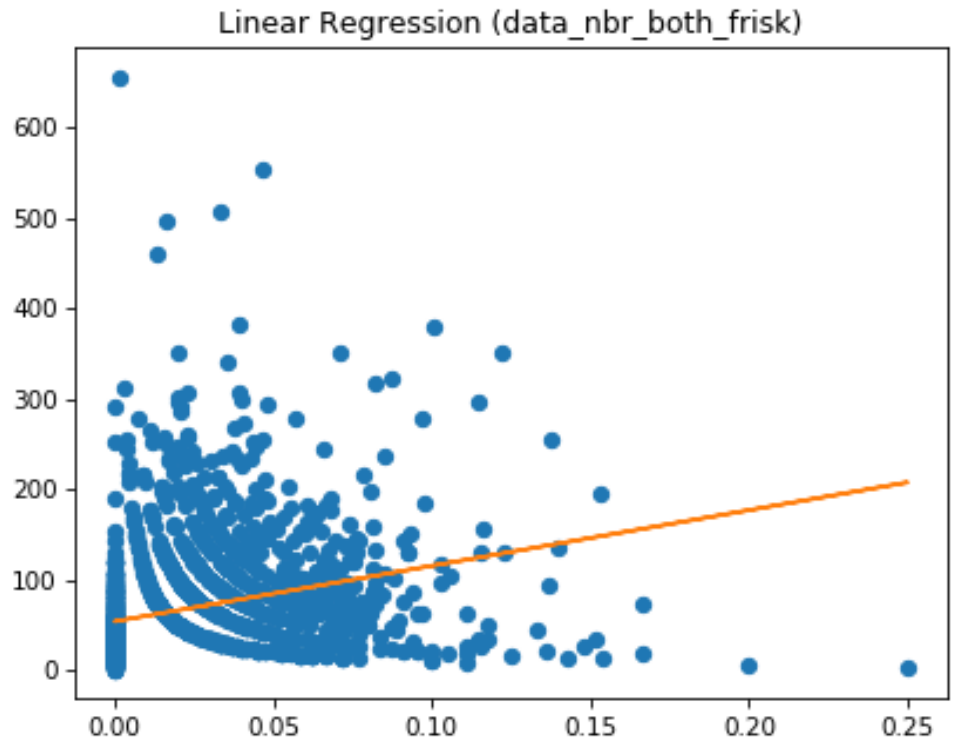
```
Pearson correlation coefficient:  0.2477298425859926
```



In [512]:

```
s1=pd.Series(data_nbr_both_frisk['total'].tolist())
s2=pd.Series(data_nbr_both_frisk['mean'].tolist())
res=s1.corr(s2)
print('Pearson correlation coefficient: ', res)

y = data_nbr_both_frisk[['total']]
x = data_nbr_both_frisk[['mean']]
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x, y)
%matplotlib notebook
plt.plot(x, y, 'o')
plt.plot(x, model.predict(x), linestyle="solid")
plt.title('Linear Regression (data_nbr_both_frisk)')
plt.show()
plt.savefig('res/52_pastcrimes_data_nbr_both_frisk.png')
```

```
Pearson correlation coefficient:  0.2755434710439923
```



Random Forest (Additional Spatiotemoral Features)

In [528]:

```
tree = pd.read_csv('7tree_num.csv')
house = pd.read_csv('6housing_density.csv')
population_income = pd.read_csv('45population_income.csv')

crime_property_census = pd.merge(crime_property, tree, left_on='Census_T
crime_property_census = pd.merge(crime_property_census, house, left_on='
crime_property_census = pd.merge(crime_property_census, population_incom
crime_property_census = pd.merge(crime_property_census, table_census_tab
crime_property_census = pd.merge(crime_property_census, pro_30D.astype({

crime_personal_census = pd.merge(crime_personal, tree, left_on='Census_T
crime_personal_census = pd.merge(crime_personal_census, house, left_on='
crime_personal_census = pd.merge(crime_personal_census, population_incom
crime_personal_census = pd.merge(crime_personal_census, table_census_tab
crime_personal_census = pd.merge(crime_personal_census, per_30D.astype({

crime_both_census = pd.merge(crime_both, tree, left_on='Census_Tracts',
crime_both_census = pd.merge(crime_both_census, house, left_on='Census_T
crime_both_census = pd.merge(crime_both_census, population_income, left_
crime_both_census = pd.merge(crime_both_census, table_census_table, left
crime_both_census = pd.merge(crime_both_census, both_30D.astype({'Census
```

In [533]:

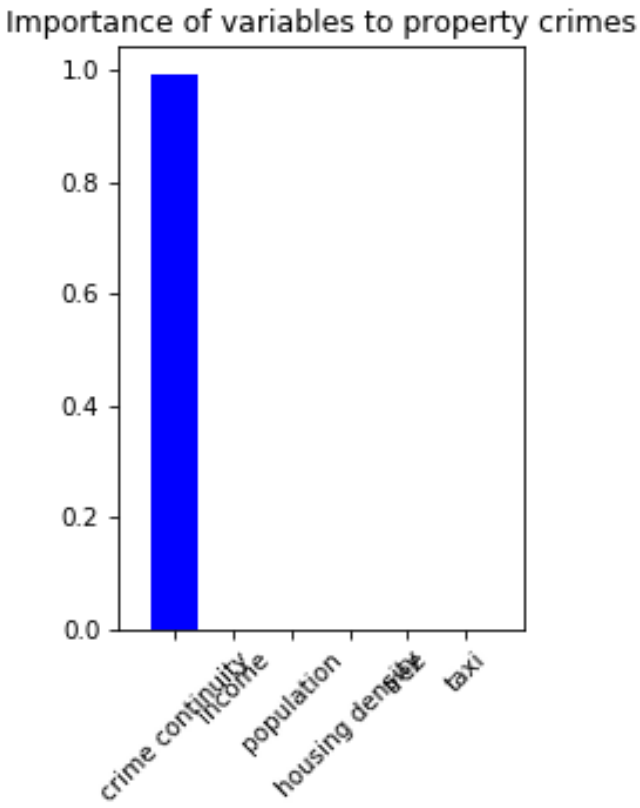
```
dropField = ['Census_Tracts', 'CTID', 'ctid', 'shape_area', 'building_a'
crime_property_census.drop(dropField, axis=1).dropna(inplace=True)
crime_personal_census.drop(dropField, axis=1).dropna(inplace=True)
crime_both_census.drop(dropField, axis=1).dropna(inplace=True)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-533-bfb102d3e74b> in <module>()
      1 dropField = ['Census_Tracts', 'CTID', 'ctid', 'shape_area', 'building_a',
'CT_ID', 'FID', 'boro_ct2010', 'TaxiZoneID', 'borough', 'Taxi_Zone']
----> 2 crime_property_census.drop(dropField, axis=1).dropna(inplace=True)
      3 crime_personal_census.drop(dropField, axis=1).dropna(inplace=True)
      4 crime_both_census.drop(dropField, axis=1).dropna(inplace=True)

~/.pyenv/versions/anaconda3-5.3.1/lib/python3.7/site-packages/pandas/core/frame.py
in drop(self, labels, axis, index, columns, level, inplace, errors)
   3995             level=level,
   3996             inplace=inplace,
-> 3997             errors=errors,
   3998         )
   3999

~/.pyenv/versions/anaconda3-5.3.1/lib/python3.7/site-packages/pandas/core/generic.
py in drop(self, labels, axis, index, columns, level, inplace, errors)
   3934         for axis, labels in axes.items():
   3935             if labels is not None:
-> 3936                 obj = obj._drop_axis(labels, axis, level=level, errors=err
ors)
   3937
   3938         if inplace:

~/.pyenv/versions/anaconda3-5.3.1/lib/python3.7/site-packages/pandas/core/generic.
py in _drop_axis(self, labels, axis, level, errors)
   3968                 new_axis = axis.drop(labels, level=level, errors=errors)
   3969             else:
-> 3970                 new_axis = axis.drop(labels, errors=errors)
   3971             result = self.reindex(**{axis_name: new_axis})
   3972

~/.pyenv/versions/anaconda3-5.3.1/lib/python3.7/site-packages/pandas/core/indexes/
base.py in drop(self, labels, errors)
   5016         if mask.any():
   5017             if errors != "ignore":
-> 5018                 raise KeyError(f"{labels[mask]} not found in axis")
   5019             indexer = indexer[~mask]
   5020         return self.delete(indexer)

KeyError: "['Census_Tracts' 'CTID' 'ctid' 'shape_area' 'building_a' 'CT_ID' 'FI
D'\n 'boro_ct2010' 'TaxiZoneID' 'borough' 'Taxi_Zone'] not found in axis"
```

In [540]:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
data_test_all = crime_property_census.sample(n=100)
data_train_all = crime_property_census.loc[list(set(crime_property_censu

y_train = data_train_all['total_x'].values.tolist()
data_train = data_train_all.drop("total_x", 1)
x_train = data_train.values.tolist()

y_test = data_test_all['total_x'].values.tolist()
data_test = data_test_all.drop("total_x", 1)
x_test = data_test.values.tolist()

rfr = RandomForestRegressor(n_estimators=1000)
rfr.fit(x_train, y_train)
predict_y = rfr.predict(x_test)
r2_score = r2_score(y_test, predict_y)
print('oefficient of determination: ', r2_score)
feature = rfr.feature_importances_
label = data_train.columns[0:]
indices = np.argsort(feature)[::-1]
for i in range(len(feature)):
    print(str(i + 1) + "   " +
          str(label[indices[i]]) + "   " + str(feature[indices[i]]))

%matplotlib notebook
plt.subplot(122, facecolor='white')
plt.title('Importance of variables to property crimes')
plt.bar(
    range(
        len(feature)),
    feature[indices],
    color='blue',
    align='center')
# plt.xticks(range(len(feature)),label[indices], rotation=45)
plt.xticks(range(len(feature)),['crime continuity', 'income', 'populatio
plt.xlim([-1, len(feature)])
plt.tight_layout()
plt.show()
plt.savefig('res/53_property_RF_2.png')
```

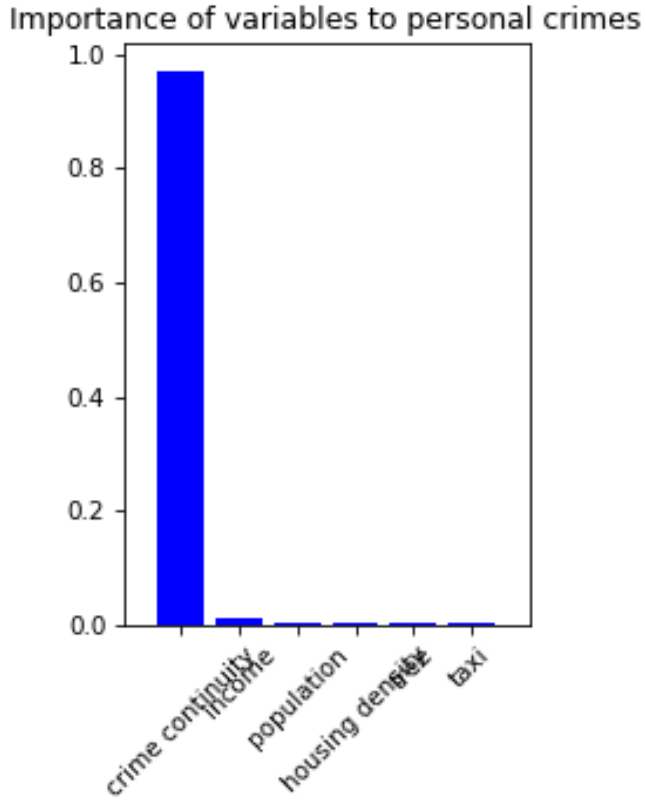
```
oefficient of determination:  0.990839235274252
1   mean   0.9903436302773033
2   Population   0.002385927713820449
3   Household_Income   0.0022965485342781302
4   NUMPOINTS   0.001746286007865662
5   total_y   0.001659452111486006
6   percentage   0.0015681553552465966
```



In [545]:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
data_test_all = crime_personal_census.sample(n=100)
data_train_all = crime_personal_census.loc[list(set(crime_personal_censu

y_train = data_train_all['total_x'].values.tolist()
data_train = data_train_all.drop("total_x", 1)
x_train = data_train.values.tolist()

y_test = data_test_all['total_x'].values.tolist()
data_test = data_test_all.drop("total_x", 1)
x_test = data_test.values.tolist()

rfr = RandomForestRegressor(n_estimators=1000)
rfr.fit(x_train, y_train)
predict_y = rfr.predict(x_test)
r2_score = r2_score(y_test, predict_y)
print('oefficient of determination: ', r2_score)
feature = rfr.feature_importances_
label = data_train.columns[0:]
indices = np.argsort(feature)[::-1]
for i in range(len(feature)):
    print(str(i + 1) + "   " +
          str(label[indices[i]]) + "   " + str(feature[indices[i]]))

%matplotlib notebook
plt.subplot(122, facecolor='white')
plt.title('Importance of variables to personal crimes')
plt.bar(
    range(
        len(feature)),
    feature[indices],
    color='blue',
    align='center')
# plt.xticks(range(len(feature)),label[indices], rotation=45)
plt.xticks(range(len(feature)),['crime continuity', 'income', 'populatio
plt.xlim([-1, len(feature)])
plt.tight_layout()
plt.show()
plt.savefig('res/54_personal_RF_2.png')
```

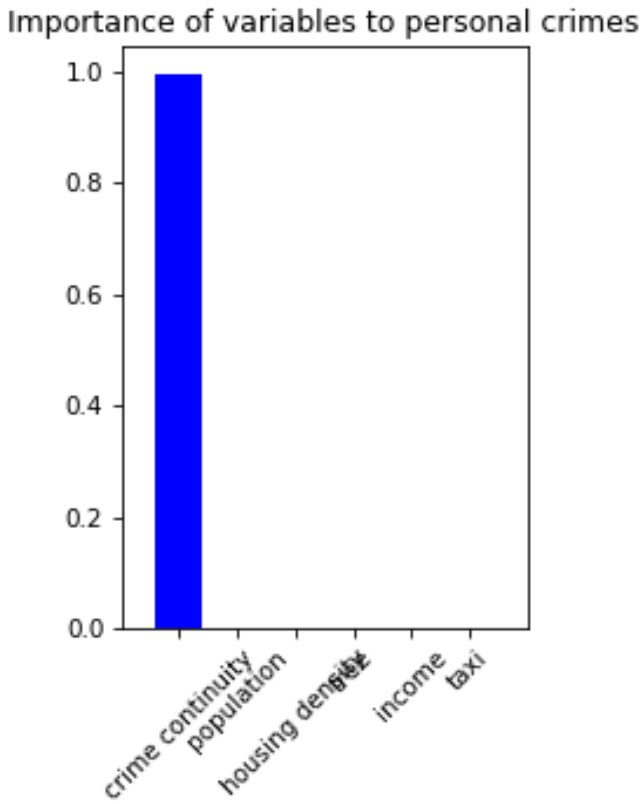
```
oefficient of determination:  0.9477045376498526
1   mean   0.9689452107245762
2   Household_Income   0.011816996915858451
3   Population   0.0061009996129752385
4   percentage   0.0047205455915356395
5   NUMPOINTS   0.004383616329886319
6   total_y   0.004032630825168056
```



In [548]:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
data_test_all = crime_both_census.sample(n=100)
data_train_all = crime_both_census.loc[list(set(crime_both_census.index.

y_train = data_train_all['total_x'].values.tolist()
data_train = data_train_all.drop("total_x", 1)
x_train = data_train.values.tolist()

y_test = data_test_all['total_x'].values.tolist()
data_test = data_test_all.drop("total_x", 1)
x_test = data_test.values.tolist()

rfr = RandomForestRegressor(n_estimators=1000)
rfr.fit(x_train, y_train)
predict_y = rfr.predict(x_test)
r2_score = r2_score(y_test, predict_y)
print('oefficient of determination: ', r2_score)
feature = rfr.feature_importances_
label = data_train.columns[0:]
indices = np.argsort(feature)[::-1]
for i in range(len(feature)):
    print(str(i + 1) + "   " +
          str(label[indices[i]]) + "   " + str(feature[indices[i]]))

%matplotlib notebook
plt.subplot(122, facecolor='white')
plt.title('Importance of variables to personal/property crimes')
plt.bar(
    range(
        len(feature)),
    feature[indices],
    color='blue',
    align='center')
# plt.xticks(range(len(feature)),label[indices], rotation=45)
plt.xticks(range(len(feature)),['crime continuity', 'population', 'housi
plt.xlim([-1, len(feature)])
plt.tight_layout()
plt.show()
plt.savefig('res/55_property_RF_2.png')
```

```
oefficient of determination:  0.9897550733167599
1   mean   0.9942103791130888
2   percentage   0.0015157623887442995
3   Population   0.001365452445190457
4   NUMPOINTS   0.0009885367373907543
5   total_y   0.0009643238007414582
6   Household_Income   0.00095554551484314
```



☝ Note: property/personal crimes

Obviously, crime continuity is a feature which strongly give an effect on the correlatioon explanation among all the explanatory variables

```
In [ ]:
```