LINEAR REGRESSION

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
import pylab
import scipy.stats as stats
import seaborn as sns
```

In [2]:
```
data = pd.read_excel("HW_Data_Set.xlsx")
data.head()
```

Out[2]:

	ind_5	ind_6	ind_8	ind_9	ind_10	ind_12	ind_13	ind_14	ind_15	ind_16	...	ind_416	ind_418	ind_420	ind_422	ind_424	ind_426	ind_42
0	19	17	100.0	85.714286	14.285714	72.363515	60.808814	23.80	17.62	11.73	...	-49.6	-54	-152	-353	1.0	0.498547	0.70190
1	24	19	100.0	78.571429	21.428571	74.275883	64.366798	11.45	18.16	12.22	...	-55.6	-60	-158	-359	1.0	0.537088	0.69083
2	30	24	100.0	71.428571	28.571429	75.140402	65.915803	8.75	17.86	12.28	...	-58.4	-60	-160	-362	1.0	0.615169	0.69304
3	37	30	100.0	64.285714	35.714286	76.677846	68.584234	7.80	14.76	12.61	...	-61.8	-65	-166	-367	1.0	0.661517	0.67341
4	41	37	100.0	57.142857	42.857143	81.603007	76.455495	14.90	11.92	14.25	...	-79.8	-86	-186	-388	1.0	0.747204	0.70052

5 rows × 136 columns

In [3]:
```
data.describe()
```

Out[3]:

	ind_5	ind_6	ind_8	ind_9	ind_10	ind_12	ind_13	ind_14	ind_15	ind_16	...	ind_412	
count	6167.000000	6167.000000	6167.000000	6167.000000	6167.000000	6167.000000	6167.000000	6167.000000	6167.000000	6167.000000	...	6167.000000	616
mean	-0.803146	-0.803470	48.388890	48.289282	0.099609	49.488867	-18.497518	11.771485	11.773550	11.773392	...	0.000224	2
std	23.624403	23.624144	36.388526	36.478009	60.414625	12.198722	68.281120	6.803997	5.786089	4.970499	...	0.005660	2
min	-131.000000	-131.000000	0.000000	0.000000	-100.000000	12.134540	-625.093855	1.050000	1.660000	2.580000	...	-0.023790	
25%	-14.000000	-14.000000	14.285714	14.285714	-50.000000	40.868503	-45.687212	6.850000	7.260000	7.635000	...	-0.003230	1
50%	0.000000	0.000000	50.000000	50.000000	0.000000	49.549766	-2.817298	10.550000	11.040000	11.270000	...	-0.000053	2
75%	13.000000	13.000000	85.714286	85.714286	57.142857	58.504375	28.072613	14.950000	15.040000	14.960000	...	0.003414	3
max	76.000000	76.000000	100.000000	100.000000	100.000000	84.821848	81.105847	82.900000	54.240000	37.050000	...	0.023340	28

8 rows × 133 columns

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6167 entries, 0 to 6166
Columns: 136 entries, ind_5 to 90_target
dtypes: float64(88), int64(45), object(3)
memory usage: 6.3+ MB
```

In [5]:
```
# Veriseti içindeki kategorik değişkenleri;

cate = data.select_dtypes(include='object')
cate
```

Out[5]:

	ind_109	ind_420	ind_422
0	GREEN	-152	-353
1	GREEN	-158	-359
2	GREEN	-160	-362
3	GREEN	-166	-367
4	GREEN	-186	-388
...	...	...	...
6162	RED	-11	-270
6163	RED	-12	-271
6164	GREEN	-21	-280
6165	GREEN	-33	-292
6166	GREEN	-28	-288

6167 rows × 3 columns

In [58]:
```
"""Veri Seti X ve y Değişkenlerine ayrılıp, X değişkeni içindeki kategorik değişkenler için

(ind_109  - GREEN/RED) dummy uygulandı """

X = dataset.iloc[:,0:133]
y = dataset.iloc[:,133]

X = pd.get_dummies(X)
```

In [73]:
```
X.tail()
```

Out[73]:

ind_12	ind_13	ind_14	ind_15	ind_16	...	ind_416	ind_418	ind_420	ind_422	ind_424	ind_426	ind_428	ind_109_-153	ind_109_GREEN	ind_109_RED
30.010531	32.362582	9.70	9.38	13.66	...	-28.2	-40	-11	-270	0.270270	0.840000	0.600846	0	0	1
30.209998	32.914628	7.65	8.84	11.79	...	-28.6	-41	-12	-271	0.000000	0.737470	0.527673	0	0	1
34.329611	43.550592	8.95	9.06	10.30	...	-37.2	-49	-21	-280	0.769231	0.632107	0.551759	0	1	0
39.027764	54.130755	13.55	10.36	10.55	...	-48.9	-61	-33	-292	1.000000	0.733010	0.591584	0	1	0
35.157180	45.524973	11.45	10.16	9.68	...	-44.0	-56	-28	-288	1.000000	0.687500	0.593936	0	1	0

LINEAR MODEL

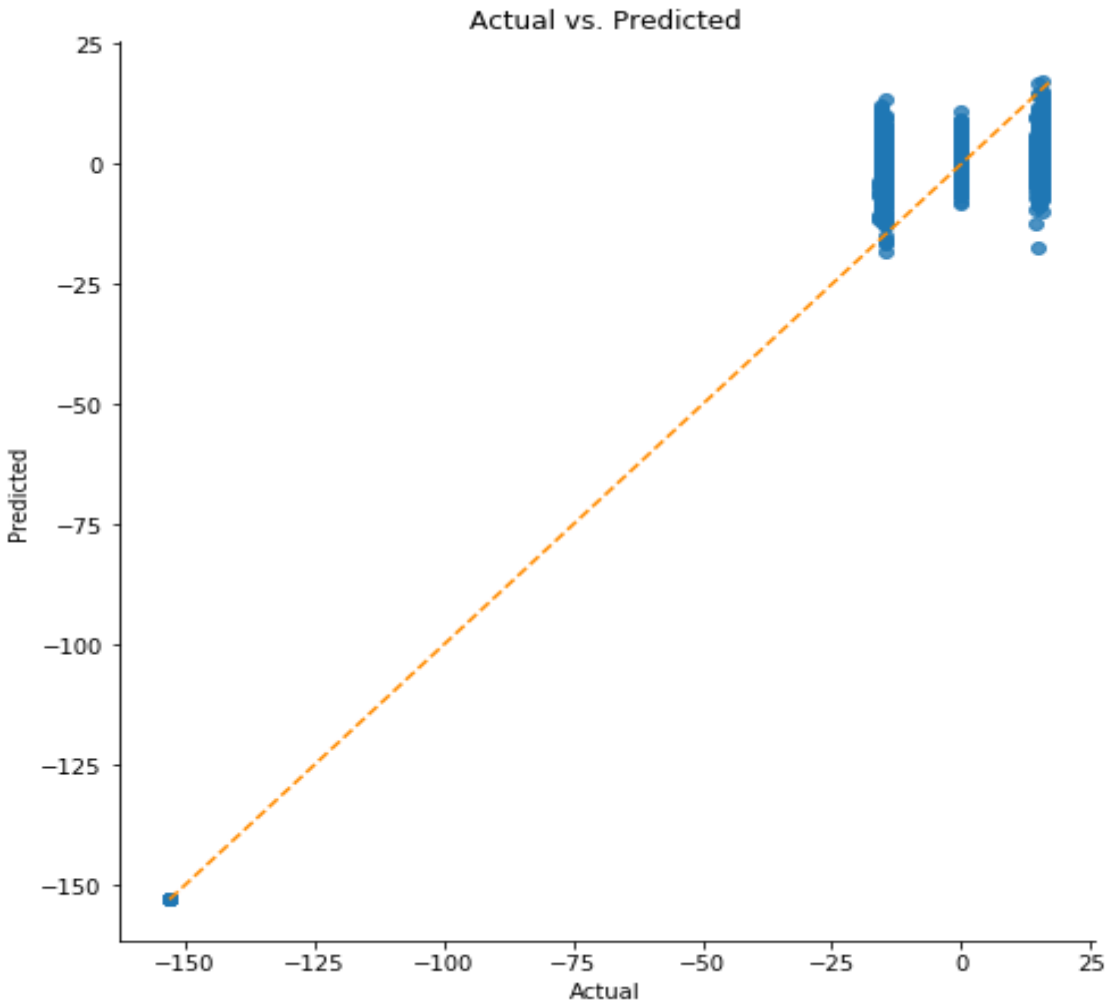
In [59]:
```
from sklearn.linear_model import LinearRegression

# Fitting the model
dataset_model = LinearRegression()
dataset_model.fit(X, y)

# Returning the R^2 for the model
data_r2 = dataset_model.score(X, y)
print('R^2: {0}'.format(data_r2))
```

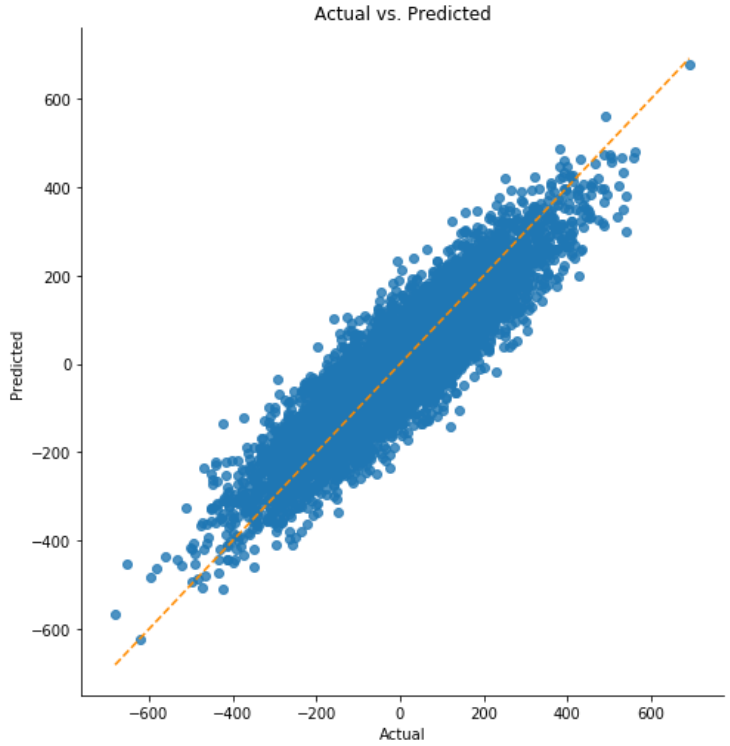
```
R^2: 0.9526885446804413
```

In [36]:
```
### STATSMODELS ###

from sklearn import datasets, linear_model
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
from scipy import stats
X1 = sm.add_constant(X)
model = sm.OLS(y, X1).fit()
predictions = model.predict(X1)

print_model = model.summary()
print(print_model)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                20_target   R-squared:                       0.953
Model:                            OLS   Adj. R-squared:                  0.952
Method:                 Least Squares   F-statistic:                     913.4
Date:                Thu, 23 Jul 2020   Prob (F-statistic):               0.00
Time:                        13:59:45   Log-Likelihood:                -24475.
No. Observations:                6167   AIC:                         4.922e+04
Df Residuals:                    6033   BIC:                         5.012e+04
Df Model:                         133
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3.171e+09   4.36e+09      0.727      0.467   -5.38e+09    1.17e+10
ind_5          -0.0770      0.339     -0.227      0.821      -0.742       0.588
ind_6           0.3160      0.276      1.146      0.252      -0.225       0.857
ind_8       -5.006e+07   5.33e+07     -0.939      0.348   -1.55e+08    5.44e+07
```

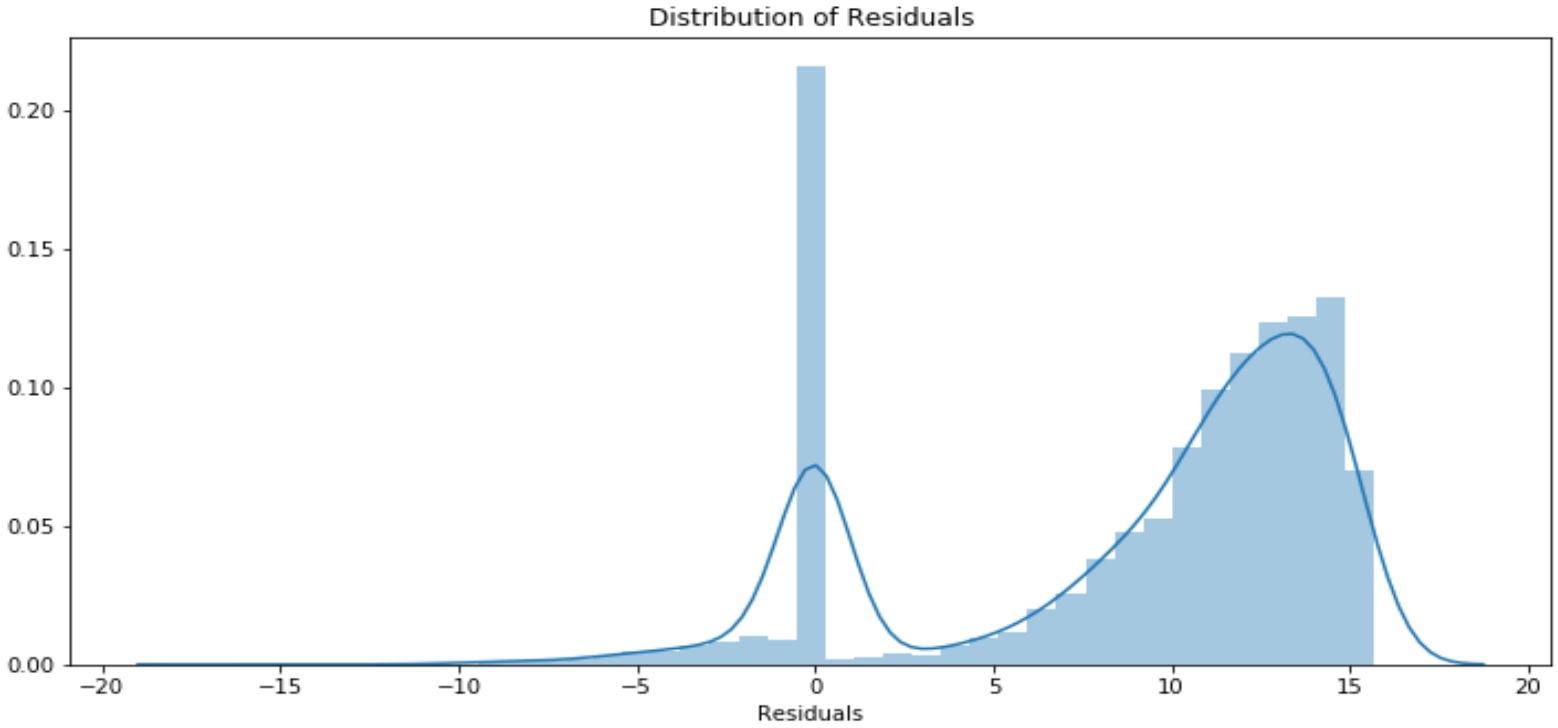
VARSAYIMLAR: DOĞRUSALLIK



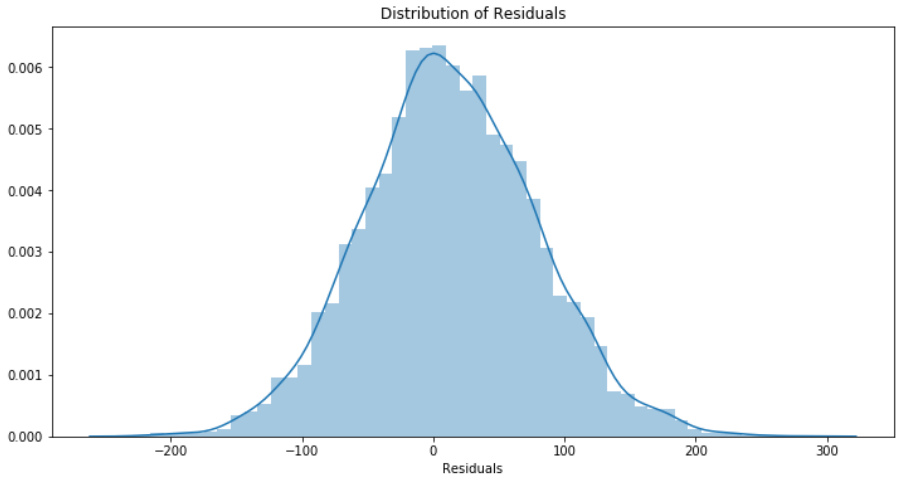


Lineer dağılıma sahip örnek bir plot.
Tahminler çizgiyi takip etmeli.

VARSAYIMLAR: HATALARIN NORMALLİĞİ



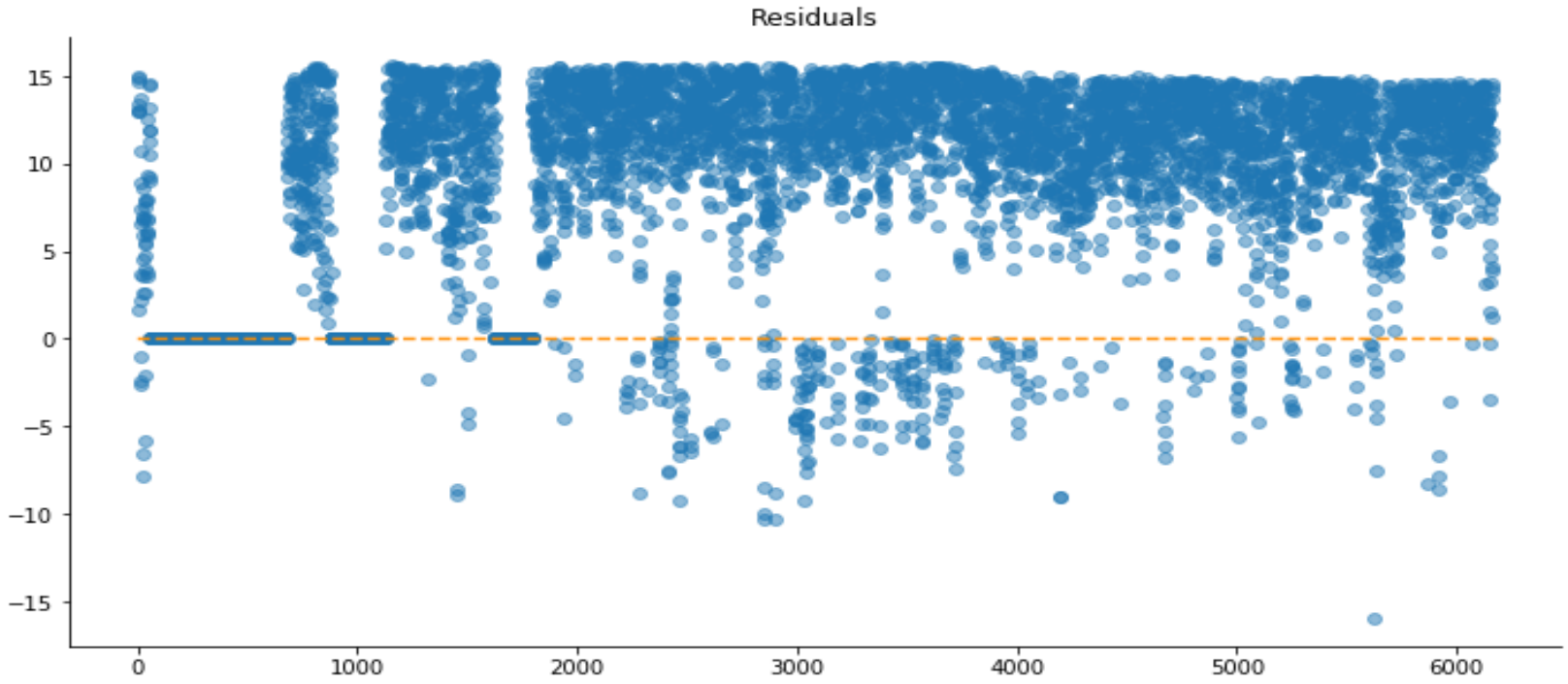
→ Bizim modelimizin grafiği



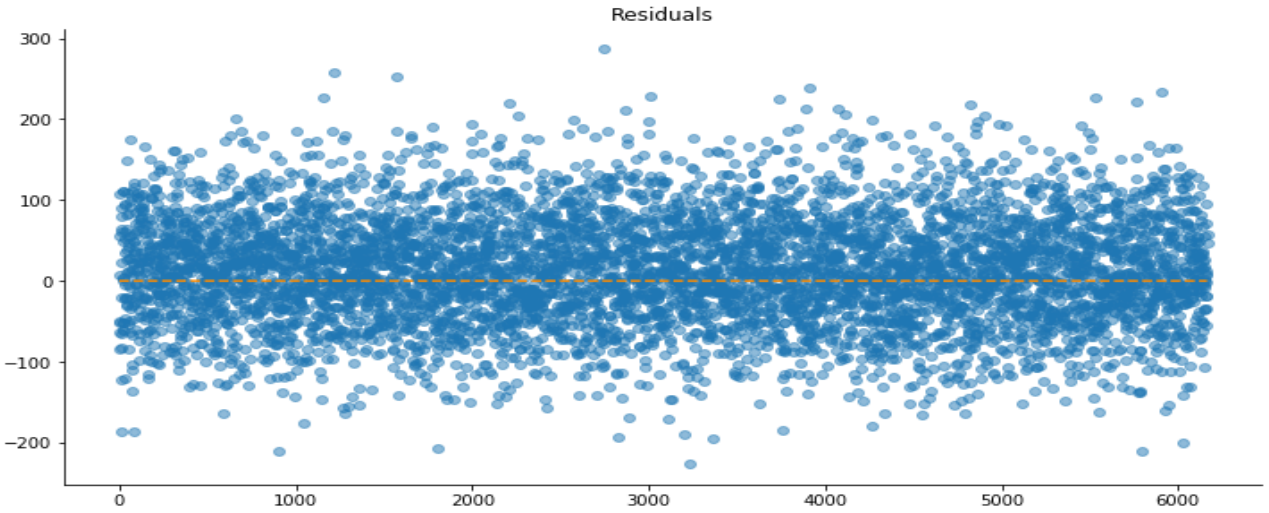
→ Normal dağılım gösteren örnek bir grafik

VARSAYIMLAR: HOMOSCEDASTİCİTY

Residuals should have relative constant variance



Residual değerler arasında tamamen tekdüze bir sapma görünmüyor,
bu nedenle model potansiyel olarak sorunlu.



İdeal bir veri setinin residual dağılımı

LINEER MODEL II

In [38]:
```
# Düşük p değerli değişkenleri atıyoruz (backward elimineted)

X2 = X.loc[:, ['ind_15','ind_17','ind_18','ind_22','ind_28','ind_30',

model2 = sm.OLS(y, X2).fit()
predictions = model2.predict(X2)

print_model = model2.summary()
print(print_model)
```

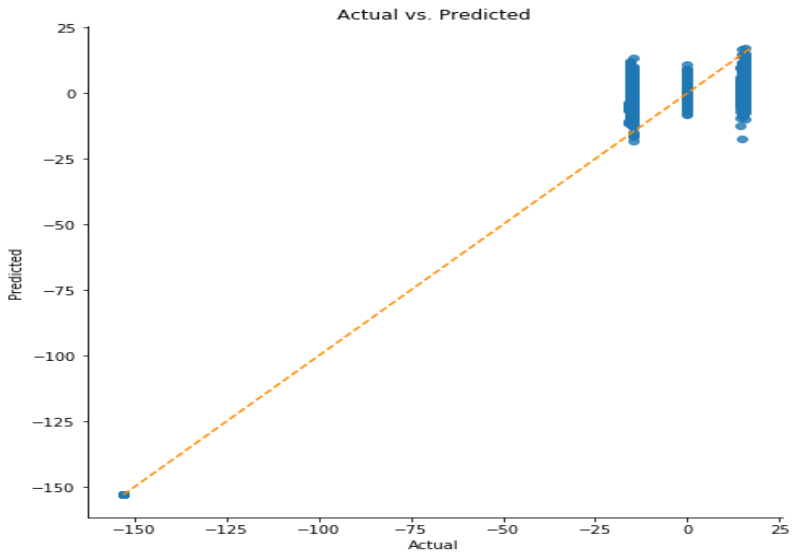
In [62]:
```
dataset_model1 = LinearRegression()
dataset_model1.fit(X2, y)

# Returning the R^2 for the model
data_r2 = dataset_model1.score(X2, y)
print('R^2: {0}'.format(data_r2))
```

```
R^2: 0.9509129356282802
```

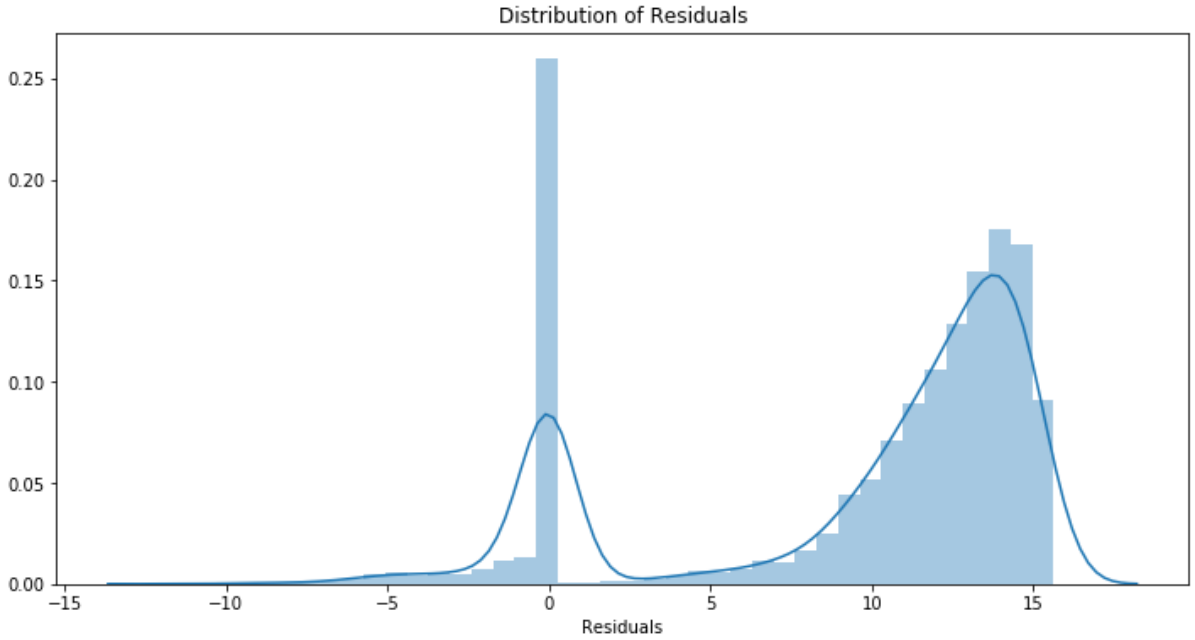
```
                                 OLS Regression Results
================================================================================================
Dep. Variable:                  20_target   R-squared (uncentered):                   0.959
Model:                              OLS   Adj. R-squared (uncentered):              0.959
Method:                   Least Squares   F-statistic:                              2970.
Date:                  Thu, 23 Jul 2020   Prob (F-statistic):                        0.00
Time:                          14:00:55   Log-Likelihood:                         -24588.
No. Observations:                  6167   AIC:                                  4.927e+04
Df Residuals:                      6119   BIC:                                  4.960e+04
Df Model:                            48
Covariance Type:            nonrobust
================================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
ind_15        -0.1850      0.050     -3.685      0.000      -0.283      -0.087
ind_17         0.6034      0.088      6.852      0.000       0.431       0.776
ind_18         0.2178      0.101      2.155      0.031       0.020       0.416
ind_22         0.1043      0.019      5.440      0.000       0.067       0.142
ind_28         0.0014      0.004      0.360      0.719      -0.006       0.009
ind_30        -0.0152      0.004     -3.409      0.001      -0.024      -0.006
ind_32        -0.0071      0.005     -1.371      0.170      -0.017       0.003
```

VARSAYIMLAR

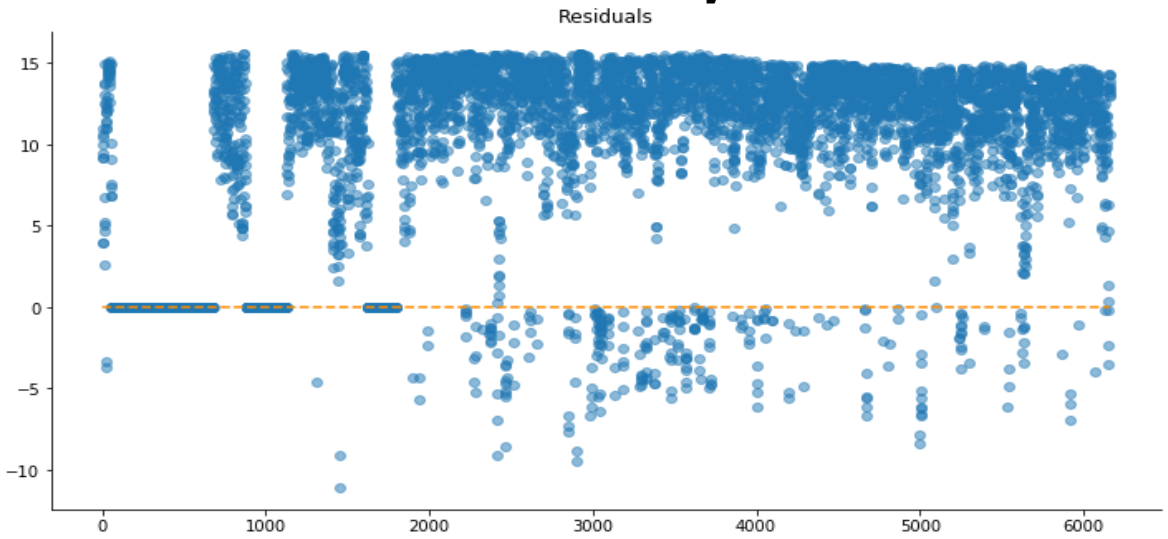
Doğrusallık



Normal Dağılım



Homoscedasticity



LINEER MODEL III

In [64]:
```
X3 = X.loc[:, ['ind_15','ind_17','ind_18','ind_22','ind_30','ind_34',

model3 = sm.OLS(y, X3).fit()
predictions = model3.predict(X3)

print_model = model3.summary()
print(print_model)
```

OLS Regression Results

Dep. Variable:	20_target	R-squared (uncentered):	0.959
Model:	OLS	Adj. R-squared (uncentered):	0.959
Method:	Least Squares	F-statistic:	4598.
Date:	Thu, 23 Jul 2020	Prob (F-statistic):	0.00
Time:	14:11:49	Log-Likelihood:	-24597.
No. Observations:	6167	AIC:	4.926e+04
Df Residuals:	6136	BIC:	4.946e+04
Df Model:	31		
Covariance Type:	nonrobust		

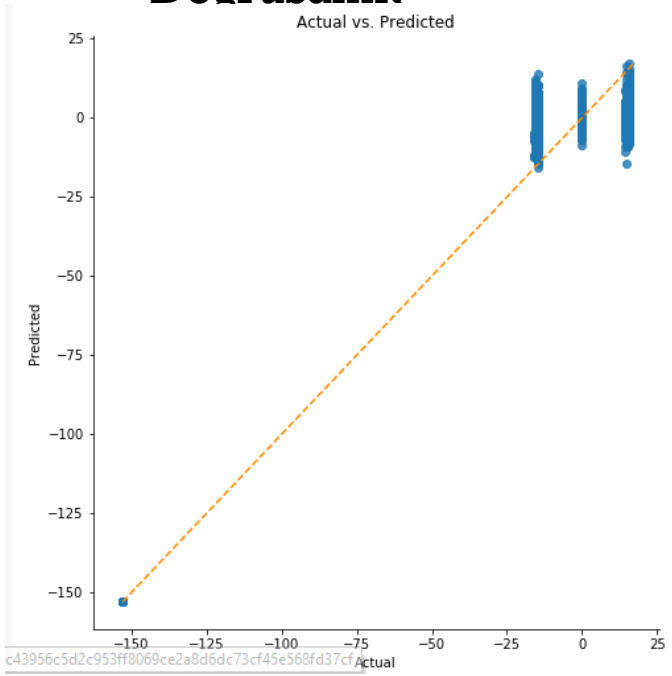
	coef	std err	t	P>|t|	[0.025	0.975]
ind_15	-0.2005	0.049	-4.104	0.000	-0.296	-0.105
ind_17	0.6422	0.085	7.540	0.000	0.475	0.809
ind_18	0.1755	0.089	1.971	0.049	0.001	0.350
ind_22	0.1039	0.018	5.681	0.000	0.068	0.140
ind_30	-0.0147	0.004	-4.014	0.000	-0.022	-0.008

In [66]:
```
dataset_model2 = LinearRegression()
dataset_model2.fit(X3, y)

# Returning the R^2 for the model
data_r2 = dataset_model2.score(X3, y)
print('R^2: {0}'.format(data_r2))
```

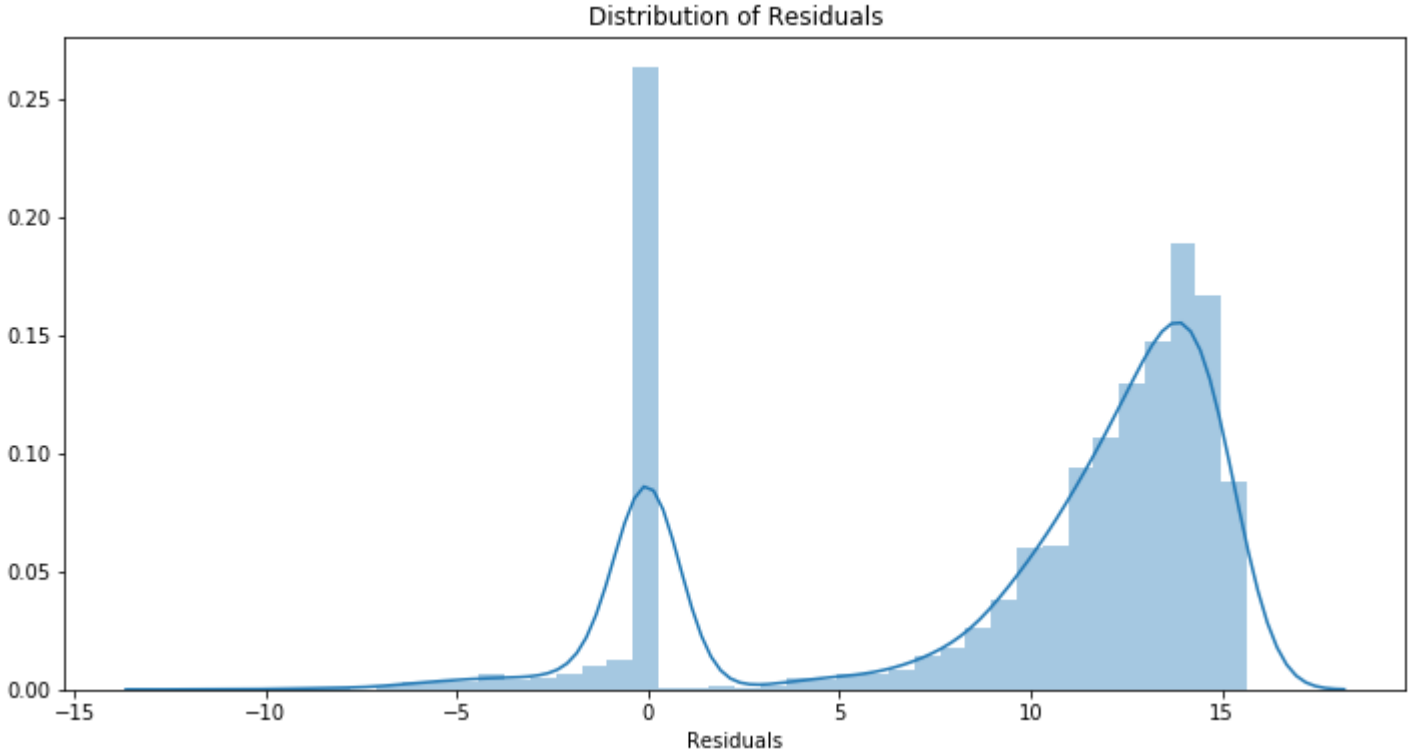
```
R^2: 0.950782989655487
```

VARSAYIMLAR

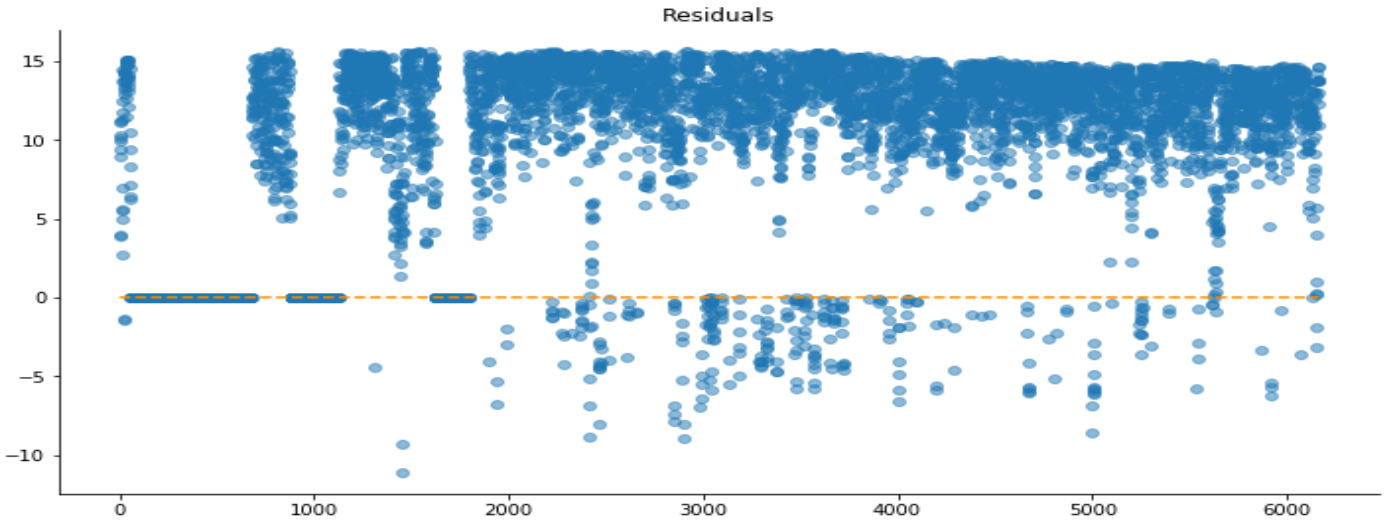
Doğrusallık



Normal Dağılım



Homoscedasticity



TRAIN – TEST SPLIT

In [12]:
```
# Train - Test Split  Model 1

from sklearn import metrics
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size = 0.3, random_state=41)

# Instantiate model
lm1 = LinearRegression()

# Fit Model
lm1.fit(X_train, y_train)

# Predict
y_pred = lm1.predict(X_test)

# RMSE
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
13.379902996452612
```

RMSE : 13.379902996452612

In [21]:
```
# Train - Test Split  Model 2

from sklearn import metrics
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X2, y,test_size = 0.3, random_state=41)

# Instantiate model
lm1 = LinearRegression()

# Fit Model
lm1.fit(X_train, y_train)

# Predict
y_pred = lm1.predict(X_test)

# RMSE
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
13.246596808891518
```

RMSE : 13.246596808891518

In [22]:
```
# Train - Test Split  Model 3

from sklearn import metrics
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X3, y,test_size = 0.3, random_state=41)

# Instantiate model
lm1 = LinearRegression()

# Fit Model
lm1.fit(X_train, y_train)

# Predict
y_pred = lm1.predict(X_test)

# RMSE
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
13.222746489647099
```

RMSE : 13.222746489647099