Polynomial logistic regression -ipo dataset

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
import statsmodels.api as sm            # sm.GLM
```

In [3]:
```
data = pd.read_csv('ipo.csv')
```

In [4]:
```
data.shape
```

Out[4]:
```
(482, 4)
```

In [5]:
```
data[:5]
```

Out[5]:

	funding	fvalue	shares	buyout
0	0	1200000	3000000	0
1	0	1454000	1454000	1
2	0	1500000	300000	0
3	0	1530000	510000	0
4	0	2000000	800000	0

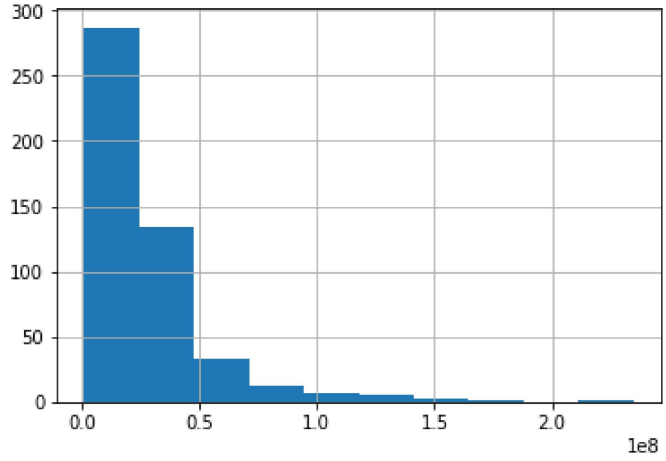
In [ ]:
```
# rows by category
```

In [8]:
```
pd.value_counts(data.funding)
```

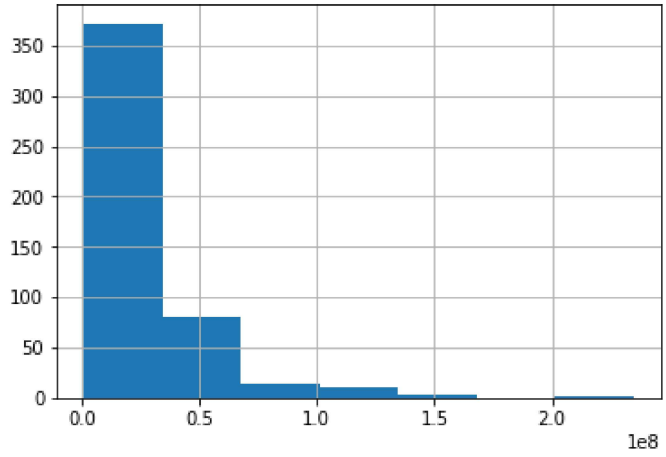
Out[8]:
```
0    270
1    212
Name: funding, dtype: int64
```

In [ ]:
```
# histogram with pandas
```

In [9]:
```
data.fvalue.hist();
```



In [10]:
```
data.fvalue.hist(bins=7);
```



In [ ]:
```
# histogram with matplotlib
```

In [11]:
```
array = data.values
```

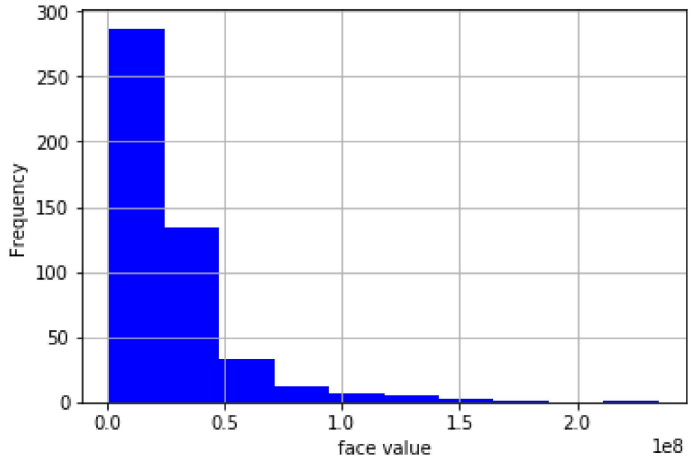
In [12]:
```
array[:5]
```

Out[12]:
```
array([[      0, 1200000, 3000000,       0],
       [      0, 1454000, 1454000,       1],
       [      0, 1500000,  300000,       0],
       [      0, 1530000,  510000,       0],
       [      0, 2000000,  800000,       0]], dtype=int64)
```

In [13]:
```
funding = array[:,0]       # target
```

In [14]:
```
fvalue  = array[:,1]       # feature
```

In [15]:
```
plt.hist(fvalue,color = "b")
plt.xlabel("face value")
plt.ylabel("Frequency")
plt.grid()
```



In [ ]:
```
# log transformation
```

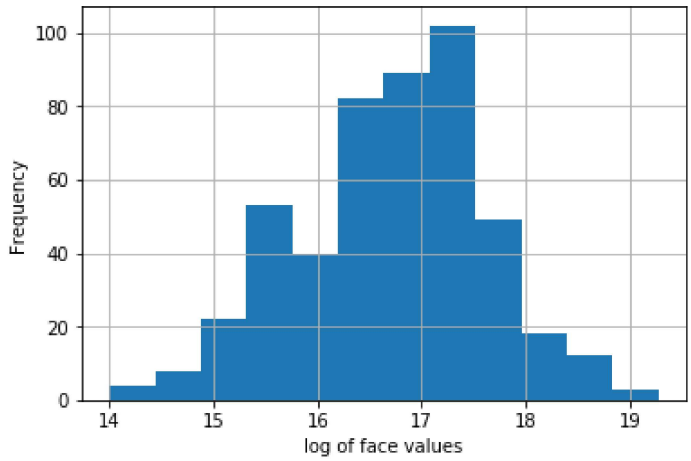
In [16]:
```
log_fvalues = np.log(array[:,1])
```

In [17]:
```
logfvalue = np.log(data["fvalue"])
```

In [18]:
```
type(logfvalue)
```

Out[18]: pandas.core.series.Series

In [19]:
```
plt.hist(logfvalue,bins=12)
plt.xlabel("log of face values")
plt.ylabel("Frequency")
plt.grid()
```



sm.GLM

In [20]:
```
x_log_fvalues = sm.add_constant(log_fvalues) # add constant
```

In [21]:
```
model_results = sm.GLM(funding,x_log_fvalues,family=sm.families.Binomial()).fit()
model_results.summary()
```

Out[21]:

Generalized Linear Model Regression Results

Dep. Variable:	y	**No. Observations:**	482
Model:	GLM	**Df Residuals:**	480
Model Family:	Binomial	**Df Model:**	1
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-321.57
Date:	Wed, 02 Oct 2019	**Deviance:**	643.13
Time:	11:09:34	**Pearson chi2:**	479.
No. Iterations:	4	**Covariance Type:**	nonrobust

	coef	std err	z	P>|z|	[0.025	0.975]
const	-7.6722	1.804	-4.253	0.000	-11.208	-4.136
x1	0.4441	0.108	4.130	0.000	0.233	0.655

In [34]:
```
b0 = model_results.params[0]
```

In [35]:
```
b1 = model_results.params[1]
```

In [22]:
```
# deviance
```

In [23]:
```
model_results.null_deviance
```

Out[23]:
```
661.1976876247234
```

In [24]:
```
model_results.deviance
```

Out[24]:
```
643.1320561968481
```

In [ ]:
```
# not much smaller than null_deviance -not good model
```

In [25]:
```
model_results.aic
```

Out[25]:
```
647.1320561968481
```

In [ ]:
```
# predicted probability row 1
```

In [26]:
```
data[:1]
```

Out[26]:

	funding	fvalue	shares	buyout
0	0	1200000	3000000	0

In [37]:
```
yhat1 = 1/(1+np.exp(-b0-b1*np.log(1200000)))
yhat1
```

Out[37]: 0.18905605986140714

In [ ]:
```
# predict probabilities all rows
```

In [ ]:
```
# model_results.predict(x_log_fvalues)[:5]    or
```

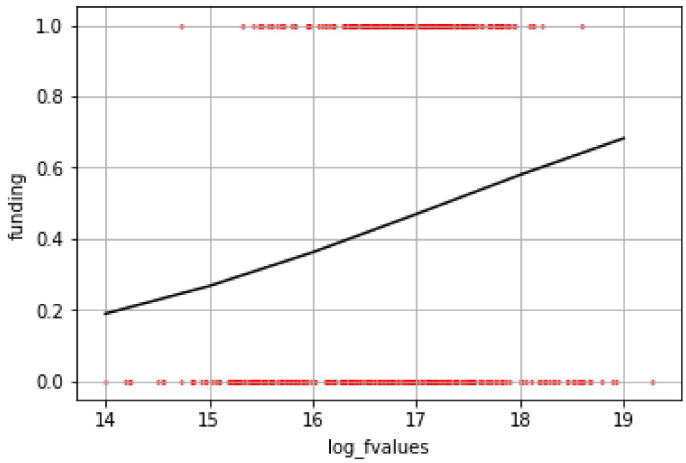
In [38]:
```
model_results.fittedvalues[:5]
```

Out[38]: array([0.18905606, 0.20247524, 0.2047179 , 0.20615332, 0.2263017])

In [39]:
```
# plot
```

In [40]:
```
xaxis = np.arange(14,19.5,1)
yaxis = model_results.predict(sm.add_constant(xaxis))
```

In [42]:
```
plt.plot(xaxis,yaxis,c='k')
plt.scatter(log_fvalues,funding,s=2,c='r')
plt.xlabel('log_fvalues')
plt.ylabel('funding')
plt.grid()
```



In [ ]:
```
# non parametric fitting -lowess
```

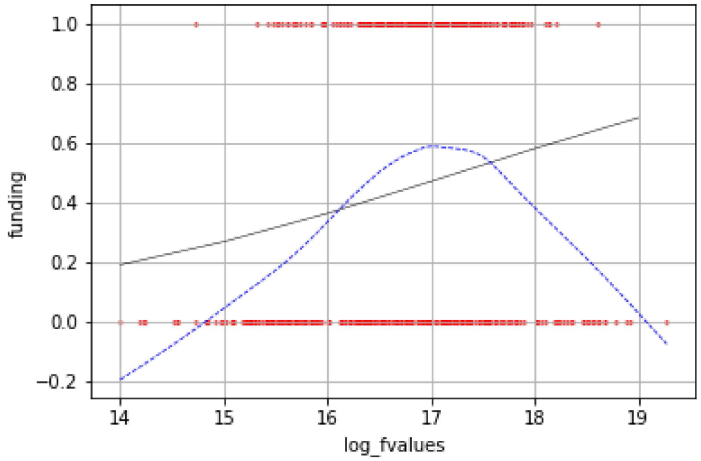
In [78]:
```
lowess = sm.nonparametric.lowess
loess = lowess(funding,log_fvalues)
```

In [79]:
```
loess_x = loess[:,0]
loess_y = loess[:,1]
```

In [80]:
```
plt.plot(xaxis,yaxis,c='k',linewidth=0.50)
plt.scatter(log_fvalues,funding,s=2,c='r')

# plot nonparametric curve
plt.plot(log_fvalues,loess_y,'--',c='b',linewidth=0.75)

plt.xlabel('log_fvalues')
plt.ylabel('funding')
plt.grid()
```



In [ ]:
```
# d) second order logistic model
```

In [81]:
```
# squared-log of face values
```

In [82]:
```
log_fvalues_2 = np.power(log_fvalues,2)
log_fvalues_2[:5]
```

Out[82]:
```
array([195.93930391, 201.35124526, 202.2361489 , 202.7997664 ,
       210.50114937])
```

In [83]:
```
input_fvalues_2 = np.vstack((log_fvalues,log_fvalues_2))
```

In [84]:
```
input_fvalues_2.shape
```

Out[84]:
```
(2, 482)
```

In [85]:
```
# transpose
```

In [86]:
```
input_fvalues_2 = np.vstack((log_fvalues,log_fvalues_2)).T
input_fvalues_2.shape
```

Out[86]:
```
(482, 2)
```

In [87]:
```
input_fvalues_2[:5]
```

Out[87]:
```
array([[ 13.99783211, 195.93930391],
       [ 14.18982894, 201.35124526],
       [ 14.22097567, 202.2361489 ],
       [ 14.24077829, 202.7997664 ],
       [ 14.50865774, 210.50114937]])
```

In [88]:
```
x_2 = sm.add_constant(input_fvalues_2)
x_2[:5]
```

Out[88]:
```
array([[  1.        ,  13.99783211, 195.93930391],
       [  1.        ,  14.18982894, 201.35124526],
       [  1.        ,  14.22097567, 202.2361489 ],
       [  1.        ,  14.24077829, 202.7997664 ],
       [  1.        ,  14.50865774, 210.50114937]])
```

In [89]:
```
model2 = sm.GLM(funding,x_2,family=sm.families.Binomial()).fit()
model2.summary()
```

Out[89]:

Generalized Linear Model Regression Results

Dep. Variable:	y	**No. Observations:**	482
Model:	GLM	**Df Residuals:**	479
Model Family:	Binomial	**Df Model:**	2
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-294.13
Date:	Wed, 02 Oct 2019	**Deviance:**	588.27
Time:	11:20:51	**Pearson chi2:**	514.
No. Iterations:	5	**Covariance Type:**	nonrobust

	coef	std err	z	P>|z|	[0.025	0.975]
const	-249.4382	39.658	-6.290	0.000	-327.167	-171.710
x1	29.3414	4.722	6.214	0.000	20.087	38.596
x2	-0.8615	0.140	-6.136	0.000	-1.137	-0.586

In [90]:
```
# deviance
```

In [91]:
```
model2.null_deviance
```

Out[91]:
```
661.1976876247234
```

In [92]:
```
model2.deviance
```

Out[92]:
```
588.2652363617015
```

In [ ]:
```
# much smaller than null deviance -good model
```

In [93]:
```
model2.aic
```

Out[93]:
```
594.2652363617015
```

In [ ]:
```
# smaller than first-order logistic model
```

In [ ]:
```
# plot 2nd order logistic fitted curve
```

In [94]:
```
xaxis = np.arange(14,19.5,0.1)   # increment is 0.10
```

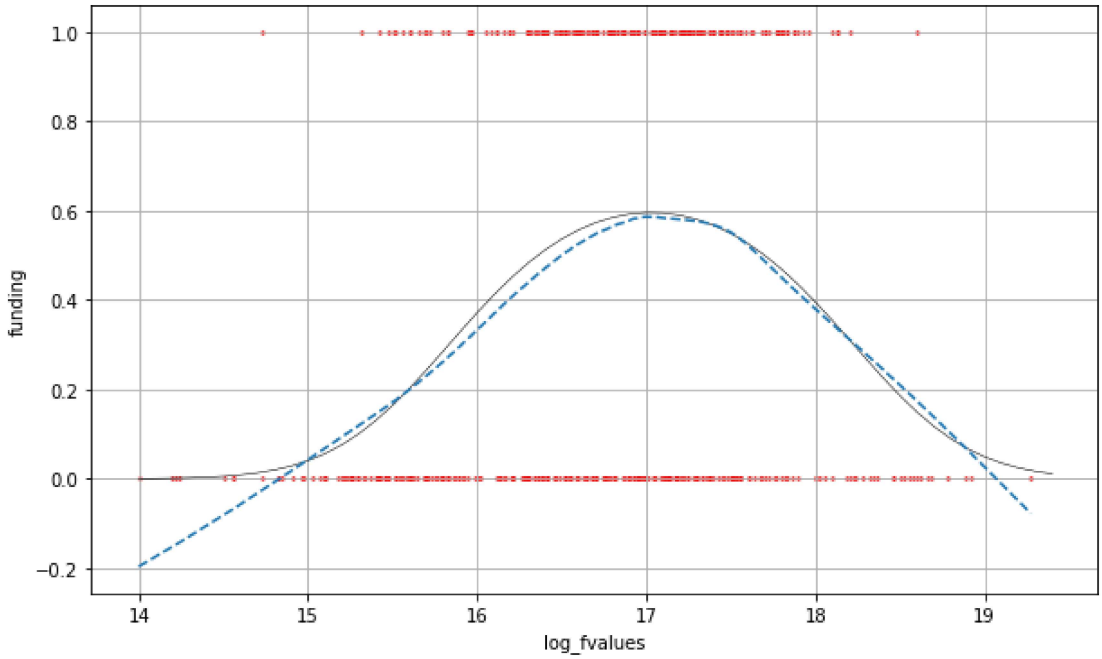
In [95]:
```
x2 = np.power(xaxis,2)
x = np.vstack((xaxis,x2)).T
```

In [96]:
```
x[:5]
```

Out[96]:
```
array([[ 14.  , 196.  ],
       [ 14.1 , 198.81],
       [ 14.2 , 201.64],
       [ 14.3 , 204.49],
       [ 14.4 , 207.36]])
```

In [97]:
```
yhat = model2.predict(sm.add_constant(x))
```

In [99]:
```
plt.figure(figsize=(10,6))
plt.plot(xaxis,yhat,c='k',linewidth=0.5)     # fitted quadratic curve
plt.scatter(log_fvalues,funding,s=2,c='r')
plt.xlabel('log_fvalues')
plt.ylabel('funding')
plt.grid()
plt.plot(log_fvalues,loess_y,'--');
```



In [100]:
```
# e) third order
```

In [101]:
```
log_fvalues_3 = np.power(log_fvalues,3)
input_fvalues_3 = np.vstack((log_fvalues,log_fvalues_2,log_fvalues_3)).T
x_3 = sm.add_constant(input_fvalues_3)
```

In [102]:
```
model_results_3 = sm.GLM(funding,x_3,family=sm.families.Binomial()).fit()
model_results_3.summary()
```

Out[102]:

Generalized Linear Model Regression Results

Dep. Variable:	y	**No. Observations:**	482
Model:	GLM	**Df Residuals:**	478
Model Family:	Binomial	**Df Model:**	3
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-293.26
Date:	Wed, 02 Oct 2019	**Deviance:**	586.52
Time:	11:22:30	**Pearson chi2:**	479.
No. Iterations:	5	**Covariance Type:**	nonrobust

	coef	std err	z	P>|z|	[0.025	0.975]
const	558.9422	581.237	0.962	0.336	-580.261	1698.145
x1	-115.4416	104.402	-1.106	0.269	-320.066	89.183
x2	7.7670	6.243	1.244	0.213	-4.469	20.003
x3	-0.1711	0.124	-1.377	0.169	-0.415	0.072

In [103]:
```
# deviance
```

In [104]:
```
model_results_3.null_deviance
```

Out[104]: 661.1976876247234

In [105]:
```
model_results_3.deviance
```

Out[105]: 586.516216123181

In [ ]:
```
# slightly smaller than deviance of second-order model
```

In [106]:
```
model_results_3.aic
```

Out[106]: 594.516216123181

In [ ]:
```
# not smaller than AIC of second-order model
```

In [ ]:
```
# 2nd order model fits better than 3rd order model
```

k-fold cross-validation -sklearn

In [108]:
```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import PolynomialFeatures
```

In [246]:
```
kfold = KFold(n_splits = 5, shuffle = True, random_state = 17)
```

In [247]:
```
log_fvalues.shape
```

Out[247]: (482,)

In [248]:
```
log_fvalues.reshape(-1,1).shape
```

Out[248]: (482, 1)

In [72]:
```
# simple logistic regression
```

In [249]:
```
np.random.seed(1)
```

In [250]:
```
model_1 = LogisticRegression(solver='lbfgs')
```

In [251]:
```
results_1 = cross_val_score(model_1,log_fvalues.reshape(-1,1),funding,cv=kfold)
```

In [252]:
```
# accuracy rate
```

In [253]:
```
arate1 = results_1.mean()
arate1
```

Out[253]: 0.5501073883161511

In [118]:
```
# loop
```

In [254]:
```
model = LogisticRegression(solver='lbfgs')
```

In [255]:
```
arate_kfold = [arate1]
```

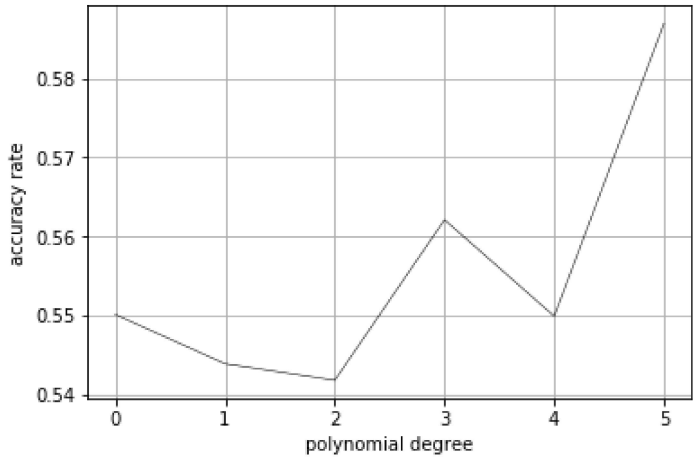
In [256]:
```
for i in range(2,7):
    form = PolynomialFeatures(degree = i)
    x = form.fit_transform(log_fvalues.reshape(-1,1))
    results = cross_val_score(model,x,funding,cv = kfold)
    arate_kfold.append(results.mean())
```

In [257]:
```
arate_kfold
```

Out[257]:
```
[0.5501073883161511,
 0.5438788659793814,
 0.5417955326460482,
 0.5620919243986254,
 0.5498926116838487,
 0.5869201030927835]
```

In [258]:
```
# plot arate
```

In [259]:
```
plt.plot(arate_kfold,c='k',linewidth=0.5)
plt.xlabel('polynomial degree')
plt.ylabel('accuracy rate')
plt.grid()
```



In [ ]:
```
# degree 3 predicts better than degree 2 logistic model
```