LO1_intro

Setup

First, let's make sure this notebook works well in both python 2 and 3, import a few common modules, ensure MatplotLib plots figures inline and prepare a function to save the figures:

In [21]:

```
#% reset -sf

# To support both python 2 and python 3
from __future__ import division, print_function, unicode_literals

# Common imports
import numpy as np
import os

# to make this notebook's output stable across runs
np.random.seed(42)

# To plot pretty figures
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
plt.rcParams['axes.labelsize'] = 14
plt.rcParams['xtick.labelsize'] = 12
plt.rcParams['ytick.labelsize'] = 12

# Where to save the figures
PROJECT_ROOT_DIR = "."
CHAPTER_ID = "fundamentals"

def save_fig(fig_id, tight_layout=True):
    path = os.path.join(PROJECT_ROOT_DIR, "images", CHAPTER_ID, fig_id + ".png")
    print("IGNORING: Saving figure", fig_id) # ITMAL: I've disabled saving of figures
    #if tight_layout:
    #    plt.tight_layout()
    #plt.savefig(path, format='png', dpi=300)

# Ignore useless warnings (see SciPy issue #5998)
import warnings
warnings.filterwarnings(action="ignore", module="scipy", message="^internal gelsd")

print("OK")
```

OK

Code example 1-1

This function just merges the OECD's life satisfaction data and the IMF's GDP per capita data. It's a bit too long and boring and it's not specific to Machine Learning, which is why I left it out of the book.

In [3]:

```
def prepare_country_stats(oecd_bli, gdp_per_capita):
    oecd_bli = oecd_bli[oecd_bli["INEQUALITY"]=="TOT"]
    oecd_bli = oecd_bli.pivot(index="Country", columns="Indicator", values="Value")
    gdp_per_capita.rename(columns={"2015": "GDP per capita"}, inplace=True)
    gdp_per_capita.set_index("Country", inplace=True)
    full_country_stats = pd.merge(left=oecd_bli, right=gdp_per_capita,
                                  left_index=True, right_index=True)
    full_country_stats.sort_values(by="GDP per capita", inplace=True)
    remove_indices = [0, 1, 6, 8, 33, 34, 35]
    keep_indices = list(set(range(36)) - set(remove_indices))
    return full_country_stats[["GDP per capita", 'Life satisfaction']].iloc[keep_indice
s]

print("OK")
```

OK

The code in the book expects the data files to be located in the current directory. I just tweaked it here to fetch the files in datasets/lifesat.

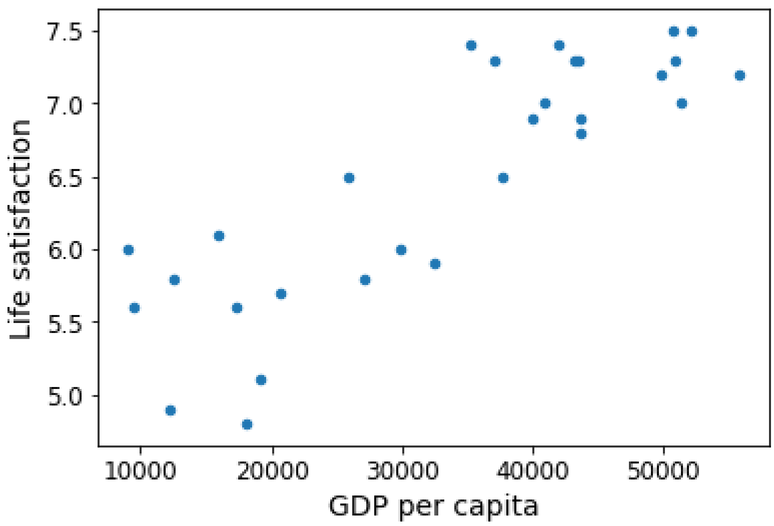
In [4]:

```
import os
datapath = os.path.join("../datasets", "lifesat", "")

print("OK")
```

OK

In [5]:

```
# Code example
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn.linear_model

# Load the data
oecd_bli = pd.read_csv(datapath + "oecd_bli_2015.csv", thousands=',')
gdp_per_capita = pd.read_csv(datapath + "gdp_per_capita.csv",thousands=',',delimiter='
\t',
                             encoding='latin1', na_values="n/a")

# Prepare the data
country_stats = prepare_country_stats(oecd_bli, gdp_per_capita)
X = np.c_[country_stats["GDP per capita"]]
y = np.c_[country_stats["Life satisfaction"]]

# Visualize the data
country_stats.plot(kind='scatter', x="GDP per capita", y='Life satisfaction')
plt.show()

# Select a linear model
model = sklearn.linear_model.LinearRegression()

# Train the model
model.fit(X, y)

# Make a prediction for Cyprus
X_new = [[22587]]  # Cyprus' GDP per capita
y_pred = model.predict(X_new)
print(y_pred) # outputs [[ 5.96242338]]

print("OK")
```




```
[[5.96242338]]
OK
```

ITMAL

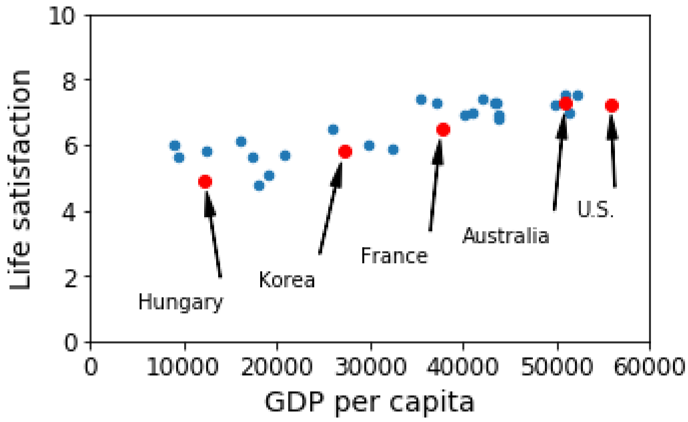
Now we plot the linear regression result.

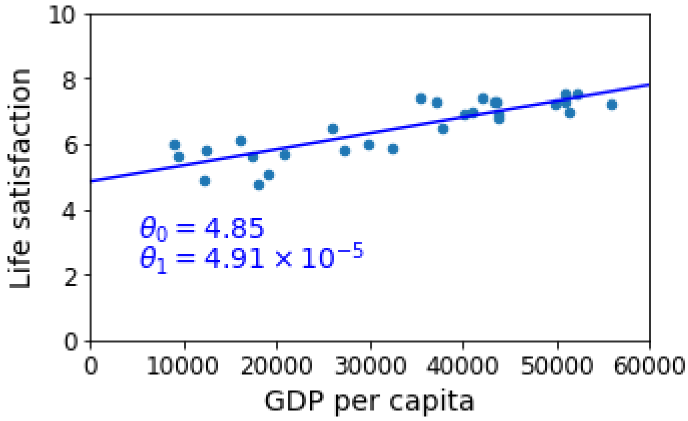
Just ignore all the data mumbo-jumbo here (from the notebook, [GITHOML])...and see the final plot.

In [6]:

```
oecd_bli = pd.read_csv(datapath + "oecd_bli_2015.csv", thousands=',')
oecd_bli = oecd_bli[oecd_bli["INEQUALITY"]=="TOT"]
oecd_bli = oecd_bli.pivot(index="Country", columns="Indicator", values="Value")
#oecd_bli.head(2)

gdp_per_capita = pd.read_csv(datapath+"gdp_per_capita.csv", thousands=',', delimiter='
\t',
                             encoding='latin1', na_values="n/a")
gdp_per_capita.rename(columns={"2015": "GDP per capita"}, inplace=True)
gdp_per_capita.set_index("Country", inplace=True)
#gdp_per_capita.head(2)

full_country_stats = pd.merge(left=oecd_bli, right=gdp_per_capita, left_index=True, rig
ht_index=True)
full_country_stats.sort_values(by="GDP per capita", inplace=True)
#full_country_stats

remove_indices = [0, 1, 6, 8, 33, 34, 35]
keep_indices = list(set(range(36)) - set(remove_indices))

sample_data = full_country_stats[["GDP per capita", 'Life satisfaction']].iloc[keep_ind
ices]
#missing_data = full_country_stats[["GDP per capita", 'Life satisfaction']].iloc[remove
_indices]

sample_data.plot(kind='scatter', x="GDP per capita", y='Life satisfaction', figsize=(5,
3))
plt.axis([0, 60000, 0, 10])
position_text = {
    "Hungary": (5000, 1),
    "Korea": (18000, 1.7),
    "France": (29000, 2.4),
    "Australia": (40000, 3.0),
    "United States": (52000, 3.8),
}
for country, pos_text in position_text.items():
    pos_data_x, pos_data_y = sample_data.loc[country]
    country = "U.S." if country == "United States" else country
    plt.annotate(country, xy=(pos_data_x, pos_data_y), xytext=pos_text,
            arrowprops=dict(facecolor='black', width=0.5, shrink=0.1, headwidth=5))
    plt.plot(pos_data_x, pos_data_y, "ro")
#save_fig('money_happy_scatterplot')
plt.show()

from sklearn import linear_model
lin1 = linear_model.LinearRegression()
Xsample = np.c_[sample_data["GDP per capita"]]
ysample = np.c_[sample_data["Life satisfaction"]]
lin1.fit(Xsample, ysample)

t0 = 4.8530528
t1 = 4.91154459e-05

sample_data.plot(kind='scatter', x="GDP per capita", y='Life satisfaction', figsize=(5,
3))
plt.axis([0, 60000, 0, 10])
M=np.linspace(0, 60000, 1000)
plt.plot(M, t0 + t1*M, "b")
plt.text(5000, 3.1, r"$\theta_0 = 4.85$", fontsize=14, color="b")
```

```
plt.text(5000, 2.2, r"$\theta_1 = 4.91 \times 10^{-5}$", fontsize=14, color="b")
#save_fig('best_fit_model_plot')
plt.show()


print("OK")
```





OK

Qa) The θ parameters and the R^2 Score

How do you extract the θ_0 and θ_1 coefficients in his life-satisfaction figure form the linear regression model?
ANSWER: Looking at the figure, it can be seen that the intercept of y-axis is given as $\theta_0 = 4.85$ and the slope is given as $\theta_1 = 4.91 \cdot 10^{-5}$.
These values are found from the linear regression of lin1, and can be extracted by using the function as, see the following code snippet.

In [7]:

```
theta_0 = lin1.coef_
theta_1 = lin1.intercept_
print("theta_0 is ", theta_0)
print("theta_1 is ", theta_1)
```

```
theta_0 is  [[4.91154459e-05]]
theta_1 is  [4.8530528]
```

Extract the score=0.734 for the model using data (X,y).
ANSWER: See code section below.

In [8]:

```
mScore = lin1.score(X,y)
print("The score of lin1 =",mScore)
```

```
The score of lin1 = 0.734441435543703
```

Explain what R^2 score measures in broad terms
ANSWER: R^2 explains how close the data are to the fitted regression line.

$$\begin{aligned} R^2 &= 1 - u/v \\ u &= \sum (y_{true} - y_{pred})^2 \qquad \text{residual sum of squares} \\ v &= \sum (y_{true} - \mu_{true})^2 \qquad \text{total sum of squares} \end{aligned}$$

with y_{true} being the true data, y_{pred} being the predicted data from the model and μ_{true} being the true mean of the data.

What are the minimum and maximum values for R^2?
ANSWER: The minimum value of R^2 is 0 and the maximum value of R^2 is 1.

Is it best to have a low R^2 score (a measure of error/loss via a cost-function) or a high R^2 score (a measure of fitness/goodness)?
ANSWER: It is best to have a high R^2 score to get the best match between data and regression line.

Qb) Using k-Nearest Neighbors

What do the k-nearest neighbours estimate for Cyprus, compared to the linear regression (it should yield=5.77)?
ANSWER: See code section below.

What *score-method* does the k-nearest model use, and is it comparable to the linear regression model?
ANSWER: k-nearest model uses R^2 scoring method. It is the same scoring method as linear regression.

In [9]:

```
# this is our raw data set:
sample_data
```

Out[9]:

	GDP per capita	Life satisfaction
Country		
Russia	9054.914	6.0
Turkey	9437.372	5.6
Hungary	12239.894	4.9
Poland	12495.334	5.8
Slovak Republic	15991.736	6.1
Estonia	17288.083	5.6
Greece	18064.288	4.8
Portugal	19121.592	5.1
Slovenia	20732.482	5.7
Spain	25864.721	6.5
Korea	27195.197	5.8
Italy	29866.581	6.0
Japan	32485.545	5.9
Israel	35343.336	7.4
New Zealand	37044.891	7.3
France	37675.006	6.5
Belgium	40106.632	6.9
Germany	40996.511	7.0
Finland	41973.988	7.4
Canada	43331.961	7.3
Netherlands	43603.115	7.3
Austria	43724.031	6.9
United Kingdom	43770.688	6.8
Sweden	49866.266	7.2
Iceland	50854.583	7.5
Australia	50961.865	7.3
Ireland	51350.744	7.0
Denmark	52114.165	7.5
United States	55805.204	7.2

In [10]:

```
# and this is our preprocessed data
country_stats
```

Out[10]:

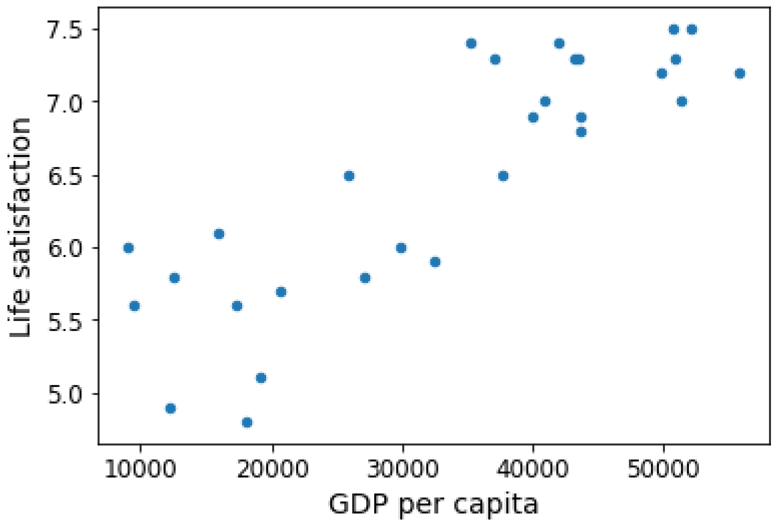
	GDP per capita	Life satisfaction
Country		
Russia	9054.914	6.0
Turkey	9437.372	5.6
Hungary	12239.894	4.9
Poland	12495.334	5.8
Slovak Republic	15991.736	6.1
Estonia	17288.083	5.6
Greece	18064.288	4.8
Portugal	19121.592	5.1
Slovenia	20732.482	5.7
Spain	25864.721	6.5
Korea	27195.197	5.8
Italy	29866.581	6.0
Japan	32485.545	5.9
Israel	35343.336	7.4
New Zealand	37044.891	7.3
France	37675.006	6.5
Belgium	40106.632	6.9
Germany	40996.511	7.0
Finland	41973.988	7.4
Canada	43331.961	7.3
Netherlands	43603.115	7.3
Austria	43724.031	6.9
United Kingdom	43770.688	6.8
Sweden	49866.266	7.2
Iceland	50854.583	7.5
Australia	50961.865	7.3
Ireland	51350.744	7.0
Denmark	52114.165	7.5
United States	55805.204	7.2

Qb Implementation

In [19]:

```
# Prepare the data
X = np.c_[country_stats["GDP per capita"]]
y = np.c_[country_stats["Life satisfaction"]]

print("X.shape=",X.shape)
print("y.shape=",y.shape)

# Visualize the data
country_stats.plot(kind='scatter', x="GDP per capita", y='Life satisfaction')
plt.show()

# Select and train a model (linear regression or k-nearest neighbours)
from sklearn.neighbors import KNeighborsRegressor
KReg = KNeighborsRegressor(n_neighbors=3)

KReg.fit(X,y)

#Predict for a specific GPD
KReg_y_pred = KReg.predict([[22587]])
#Score of regressor
KReg_score = KReg.score(X,y)

print(KReg_y_pred)
print(KReg_score)
```

```
X.shape= (29, 1)
y.shape= (29, 1)
```




```
[[5.76666667]]
0.8525732853499179
```

Qb Result

We see that the result of the prediction is 5.7. The k_nearest neighbors =3 takes the nearest three data points and calculates the average of them. This is the prediction that we see.

Qc) Tuning Parameter for k-Nearest Neighbors and A Sanity Check

Plot the two models in a 'Life Satisfaction-vs-GDP capita' 2D plot by creating an array in the range 0 to 60000 (USD) and then predict the corresponding y value. Reuse the plots stubs below, and explain why the k-nearest neighbour with `k_neighbor=1` has such a good score.
ANSWER: K-nearest neighbour uses R^2 scoring method. This scoring method tells us how close our regression line is to the data points. With `k_neighbor=1` the regression line is going from data point to data point. Therefore the regression line will hit every data point and the score will be 1.

Does a score=1 with `k_neighbor=1` also mean that this would be the prefered estimator for the job?
ANSWER: With `k_neighbor=1` the prediction would have the same value as its closest neighbor. With a higher k_neighbor value the prediction becomes the average of its k_neighbors. This means that a low k_neighbor value makes noise in the dataset a problem. With a too high k_neighbor value the prediction becomes less accurate. When using this method, it is important to find the best k_neighbor value. With `k_neighbor=1` it is most likely never the prefered estimator.

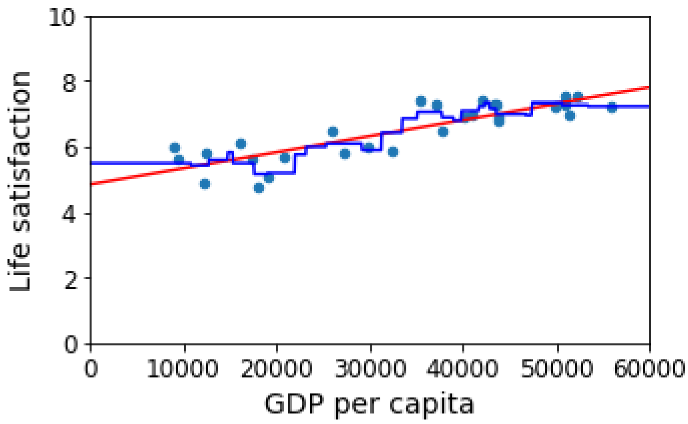
Qc Implementation

In [20]:

```
sample_data.plot(kind='scatter', x="GDP per capita", y='Life satisfaction', figsize=(5,
3))
plt.axis([0, 60000, 0, 10])

# create an test matrix M, with the same dimensionality as X, and in the range [0;6000
0]
# and a step size of your choice
m=np.linspace(0, 60000, 1000)
M=np.empty([m.shape[0],1])
M[:,0]=m

# TODO from this test M data, predict the y values via the lin.reg. and k-nearest model
s
y_pred_lin = lin1.predict(M)
y_pred_kn  = KReg.predict(M)

# TODO use plt.plot to plot x-y into the sample_data plot...
plt.plot( M, y_pred_lin , "r")
plt.plot( M, y_pred_kn , "b")
```

Out[20]:

```
[<matplotlib.lines.Line2D at 0xb749f98>]
```



Qc Result

The output above show us the two regressors. The Linear regressor is red and the k_nearest is blue. K_nearest uses 3 nearest neighbours. This means that