**Chapter 1 – The Machine Learning landscape**

*This is the code used to generate some of the figures in chapter 1.*

# 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 [73]:

```
# 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("Saving figure", fig_id)
    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")
```

# 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 [107]:

```
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_indices]
```

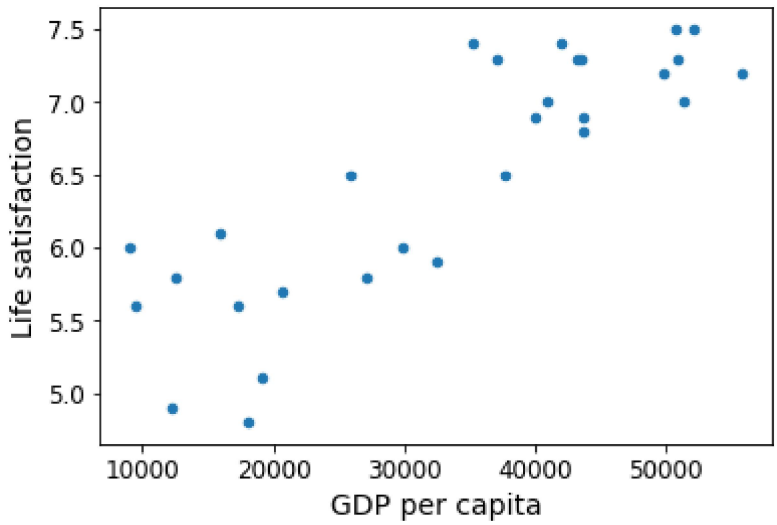
The code in the book expects the data files to be located in the current directory. I just tweaked it here to fetch the files in datasets/lifesat.

In [108]:

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

In [112]:

```
# 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
print(model.predict(X_new)) # outputs [[ 5.96242338]]
```




```
[[5.76666667]]
```

In [ ]:

In [ ]:

In [ ]:

```
```

# Note: you can ignore the rest of this notebook, it just generates many of the figures in chapter 1.

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

# Load and prepare Life satisfaction data

If you want, you can get fresh data from the OECD's website. Download the CSV from http://stats.oecd.org/index.aspx?DataSetCode=BLI (http://stats.oecd.org/index.aspx?DataSetCode=BLI) and save it to `datasets/lifesat/` .

In [77]:

```
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)
```

Out[77]:

| Indicator | Air pollution | Assault rate | Consultation on rule-making | Dwellings without basic facilities | Educational attainment | Employees working very long hours | Employment rate | H |
|---|---|---|---|---|---|---|---|---|
| **Country** | | | | | | | | |
| **Australia** | 13.0 | 2.1 | 10.5 | 1.1 | 76.0 | 14.02 | 72.0 | |
| **Austria** | 27.0 | 3.4 | 7.1 | 1.0 | 83.0 | 7.61 | 72.0 | |

2 rows × 24 columns

In [78]:

```
oecd_bli["Life satisfaction"].head()
```

Out[78]:

```
Country
Australia    7.3
Austria      6.9
Belgium      6.9
Brazil       7.0
Canada       7.3
Name: Life satisfaction, dtype: float64
```

# Load and prepare GDP per capita data

Just like above, you can update the GDP per capita data if you want. Just download data from http://goo.gl/j1MSKe (http://goo.gl/j1MSKe) (=> imf.org) and save it to `datasets/lifesat/` .

In [79]:

```
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)
```

Out[79]:

| Country | Subject Descriptor | Units | Scale | Country/Series-specific Notes | GDP per capita | Estimates Start After |
|---|---|---|---|---|---|---|
| **Afghanistan** | Gross domestic product per capita, current prices | U.S. dollars | Units | See notes for: Gross domestic product, curren... | 599.994 | 2013.0 |
| **Albania** | Gross domestic product per capita, current prices | U.S. dollars | Units | See notes for: Gross domestic product, curren... | 3995.383 | 2010.0 |

In [80]:

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

| Country | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Portugal** | 18.0 | 5.7 | 6.5 | 0.9 | 38.0 | 9.62 | 61.0 | 1.1 |
| **Slovenia** | 26.0 | 3.9 | 10.3 | 0.5 | 85.0 | 5.63 | 63.0 | 0.4 |
| **Spain** | 24.0 | 4.2 | 7.3 | 0.1 | 55.0 | 5.89 | 56.0 | 0.6 |

In [81]:

```
full_country_stats[["GDP per capita", 'Life satisfaction']].loc["United States"]
```

Out[81]:

```
GDP per capita       55805.204
Life satisfaction        7.200
Name: United States, dtype: float64
```

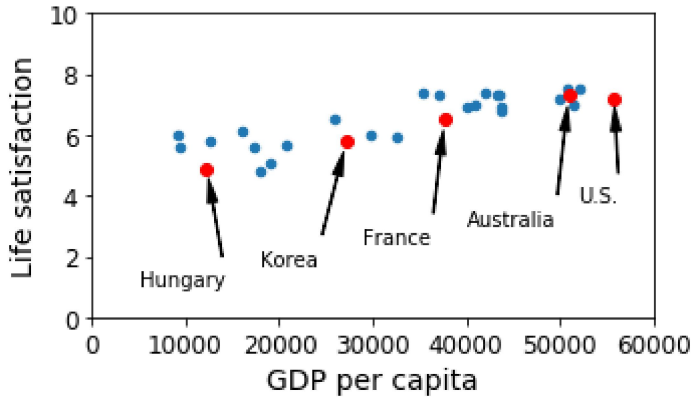
In [82]:

```
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_indices
missing_data = full_country_stats[["GDP per capita", 'Life satisfaction']].iloc[remove_indi
```

In [83]:

```
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()
```

```
Saving figure money_happy_scatterplot
```




In [84]:

```
sample_data.to_csv(os.path.join("datasets", "lifesat", "lifesat.csv"))
```

In [85]:

```
sample_data.loc[list(position_text.keys())]
```

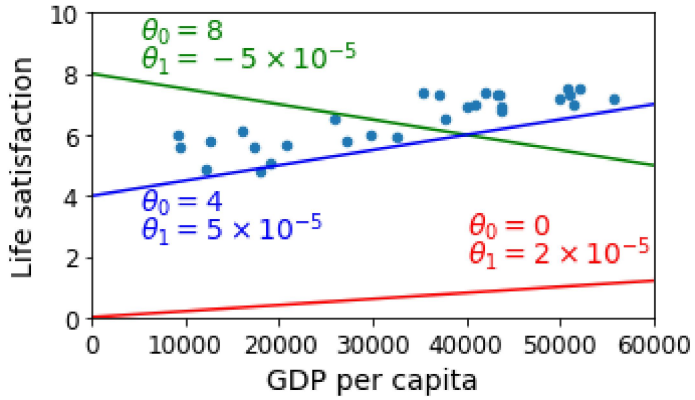
Out[85]:

| Country | GDP per capita | Life satisfaction |
|---|---|---|
| **Hungary** | 12239.894 | 4.9 |
| **Korea** | 27195.197 | 5.8 |
| **France** | 37675.006 | 6.5 |
| **Australia** | 50961.865 | 7.3 |
| **United States** | 55805.204 | 7.2 |

In [86]:

```
import numpy as np

sample_data.plot(kind='scatter', x="GDP per capita", y='Life satisfaction', figsize=(5,3))
plt.axis([0, 60000, 0, 10])
X=np.linspace(0, 60000, 1000)
plt.plot(X, 2*X/100000, "r")
plt.text(40000, 2.7, r"$\theta_0 = 0$", fontsize=14, color="r")
plt.text(40000, 1.8, r"$\theta_1 = 2 \times 10^{-5}$", fontsize=14, color="r")
plt.plot(X, 8 - 5*X/100000, "g")
plt.text(5000, 9.1, r"$\theta_0 = 8$", fontsize=14, color="g")
plt.text(5000, 8.2, r"$\theta_1 = -5 \times 10^{-5}$", fontsize=14, color="g")
plt.plot(X, 4 + 5*X/100000, "b")
plt.text(5000, 3.5, r"$\theta_0 = 4$", fontsize=14, color="b")
plt.text(5000, 2.6, r"$\theta_1 = 5 \times 10^{-5}$", fontsize=14, color="b")
save_fig('tweaking_model_params_plot')
plt.show()
```

```
Saving figure tweaking_model_params_plot
```




In [87]:

```
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, t1 = lin1.intercept_[0], lin1.coef_[0][0]
t0, t1
```

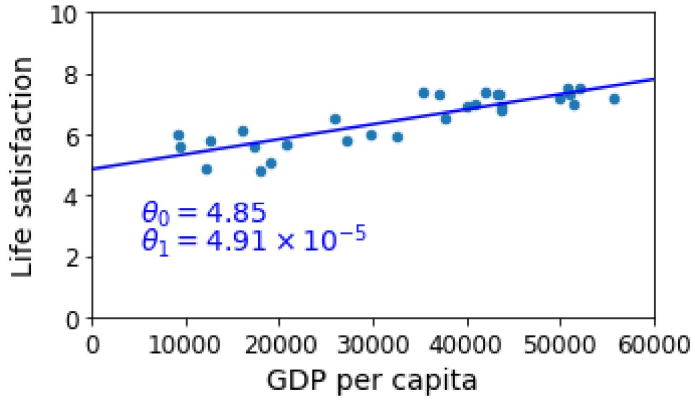
Out[87]:

```
(4.853052800266436, 4.911544589158483e-05)
```

In [88]:

```
sample_data.plot(kind='scatter', x="GDP per capita", y='Life satisfaction', figsize=(5,3))
plt.axis([0, 60000, 0, 10])
X=np.linspace(0, 60000, 1000)
plt.plot(X, t0 + t1*X, "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()
```

```
Saving figure best_fit_model_plot
```




In [89]:

```
cyprus_gdp_per_capita = gdp_per_capita.loc["Cyprus"]["GDP per capita"]
print(cyprus_gdp_per_capita)
cyprus_predicted_life_satisfaction = lin1.predict(cyprus_gdp_per_capita)[0][0]
cyprus_predicted_life_satisfaction
```

```
22587.49
```

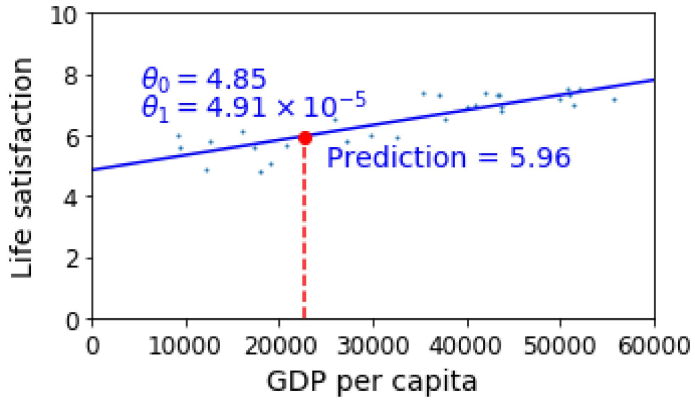
Out[89]:

```
5.96244744318815
```

In [90]:

```
sample_data.plot(kind='scatter', x="GDP per capita", y='Life satisfaction', figsize=(5,3),
X=np.linspace(0, 60000, 1000)
plt.plot(X, t0 + t1*X, "b")
plt.axis([0, 60000, 0, 10])
plt.text(5000, 7.5, r"$\theta_0 = 4.85$", fontsize=14, color="b")
plt.text(5000, 6.6, r"$\theta_1 = 4.91 \times 10^{-5}$", fontsize=14, color="b")
plt.plot([cyprus_gdp_per_capita, cyprus_gdp_per_capita], [0, cyprus_predicted_life_satisfac
plt.text(25000, 5.0, r"Prediction = 5.96", fontsize=14, color="b")
plt.plot(cyprus_gdp_per_capita, cyprus_predicted_life_satisfaction, "ro")
save_fig('cyprus_prediction_plot')
plt.show()
```

```
Saving figure cyprus_prediction_plot
```




In [91]:

```
sample_data[7:10]
```

Out[91]:

| Country | GDP per capita | Life satisfaction |
|---|---|---|
| **Portugal** | 19121.592 | 5.1 |
| **Slovenia** | 20732.482 | 5.7 |
| **Spain** | 25864.721 | 6.5 |

In [92]:

```
(5.1+5.7+6.5)/3
```

Out[92]:

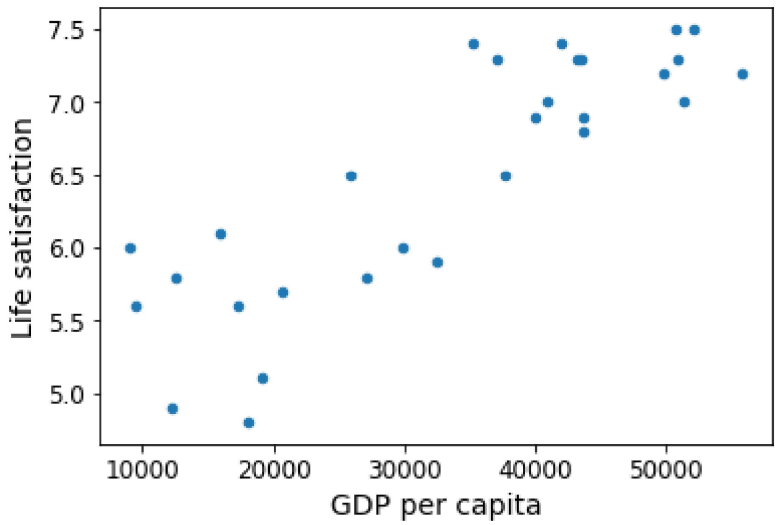
```
5.766666666666667
```

In [93]:

```
backup = oecd_bli, gdp_per_capita

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_indices]
```

In [94]:

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

# 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
print(model.predict(X_new)) # outputs [[ 5.96242338]]
```




```
[[5.96242338]]
```

In [95]:

```
oecd_bli, gdp_per_capita = backup
```

In [96]:

```
missing_data
```

Out[96]:

| Country | GDP per capita | Life satisfaction |
|---|---|---|
| **Brazil** | 8669.998 | 7.0 |
| **Mexico** | 9009.280 | 6.7 |
| **Chile** | 13340.905 | 6.7 |
| **Czech Republic** | 17256.918 | 6.5 |
| **Norway** | 74822.106 | 7.4 |
| **Switzerland** | 80675.308 | 7.5 |
| **Luxembourg** | 101994.093 | 6.9 |

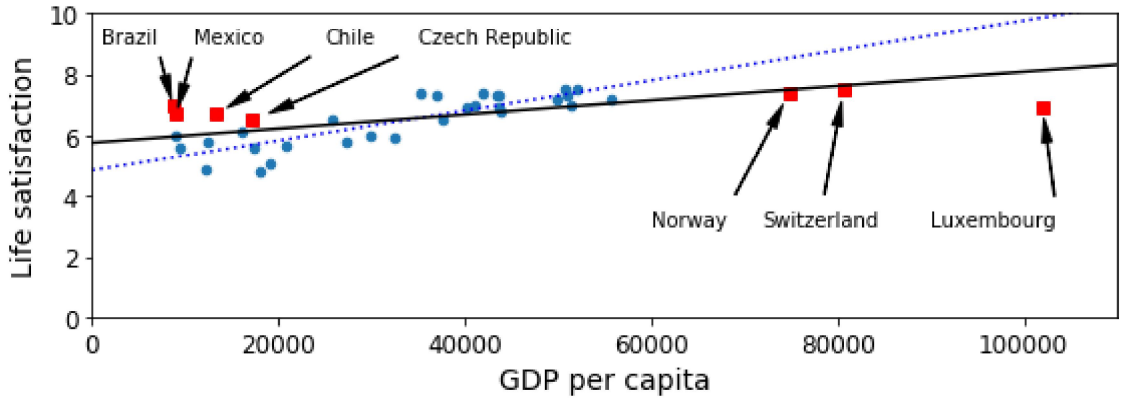
In [97]:

```
position_text2 = {
    "Brazil": (1000, 9.0),
    "Mexico": (11000, 9.0),
    "Chile": (25000, 9.0),
    "Czech Republic": (35000, 9.0),
    "Norway": (60000, 3),
    "Switzerland": (72000, 3.0),
    "Luxembourg": (90000, 3.0),
}
```

In [98]:

```
sample_data.plot(kind='scatter', x="GDP per capita", y='Life satisfaction', figsize=(8,3))
plt.axis([0, 110000, 0, 10])

for country, pos_text in position_text2.items():
    pos_data_x, pos_data_y = missing_data.loc[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, "rs")

X=np.linspace(0, 110000, 1000)
plt.plot(X, t0 + t1*X, "b:")

lin_reg_full = linear_model.LinearRegression()
Xfull = np.c_[full_country_stats["GDP per capita"]]
yfull = np.c_[full_country_stats["Life satisfaction"]]
lin_reg_full.fit(Xfull, yfull)

t0full, t1full = lin_reg_full.intercept_[0], lin_reg_full.coef_[0][0]
X = np.linspace(0, 110000, 1000)
plt.plot(X, t0full + t1full * X, "k")

save_fig('representative_training_data_scatterplot')
plt.show()
```

```
Saving figure representative_training_data_scatterplot
```

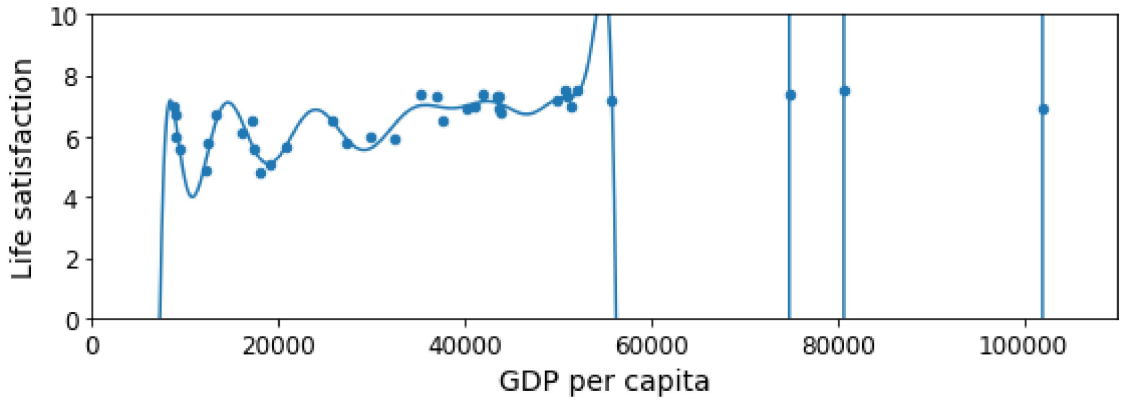


In [99]:

```
full_country_stats.plot(kind='scatter', x="GDP per capita", y='Life satisfaction', figsize=
plt.axis([0, 110000, 0, 10])

from sklearn import preprocessing
from sklearn import pipeline

poly = preprocessing.PolynomialFeatures(degree=60, include_bias=False)
scaler = preprocessing.StandardScaler()
lin_reg2 = linear_model.LinearRegression()

pipeline_reg = pipeline.Pipeline([('poly', poly), ('scal', scaler), ('lin', lin_reg2)])
pipeline_reg.fit(Xfull, yfull)
curve = pipeline_reg.predict(X[:, np.newaxis])
plt.plot(X, curve)
save_fig('overfitting_model_plot')
plt.show()
```

```
C:\Users\kofen\Anaconda\lib\site-packages\numpy\core\_methods.py:116: Runtim
eWarning: overflow encountered in multiply
  x = um.multiply(x, x, out=x)
C:\Users\kofen\Anaconda\lib\site-packages\numpy\core\_methods.py:117: Runtim
eWarning: overflow encountered in reduce
  ret = umr_sum(x, axis, dtype, out, keepdims)
```

```
Saving figure overfitting_model_plot
```




In [100]:

```
full_country_stats.loc[[c for c in full_country_stats.index if "W" in c.upper()]]["Life sat
```

Out[100]:

```
Country
New Zealand    7.3
Sweden         7.2
Norway         7.4
Switzerland    7.5
Name: Life satisfaction, dtype: float64
```

In [101]:

```
gdp_per_capita.loc[[c for c in gdp_per_capita.index if "W" in c.upper()]].head()
```

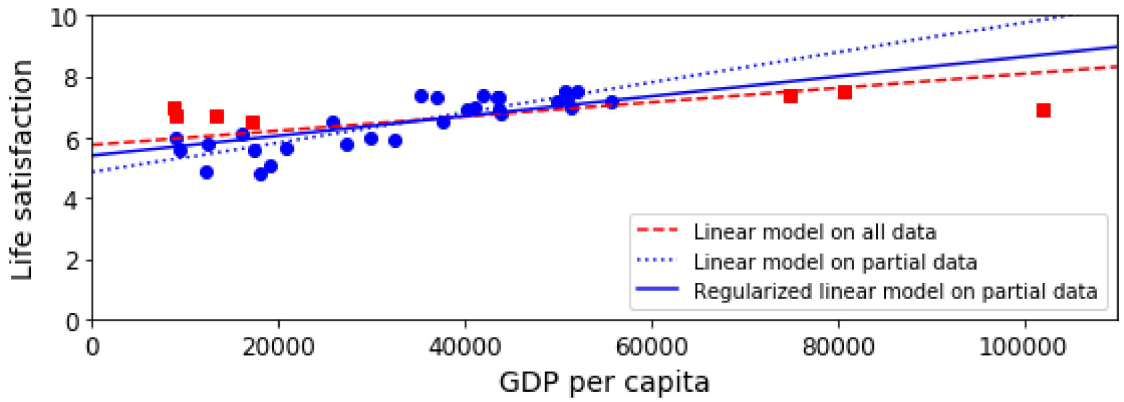
Out[101]:

| Country | Subject Descriptor | Units | Scale | Country/Series-specific Notes | GDP per capita | Estimates Start After |
|---|---|---|---|---|---|---|
| **Botswana** | Gross domestic product per capita, current prices | U.S. dollars | Units | See notes for: Gross domestic product, curren... | 6040.957 | 2008.0 |
| **Kuwait** | Gross domestic product per capita, current prices | U.S. dollars | Units | See notes for: Gross domestic product, curren... | 29363.027 | 2014.0 |
| **Malawi** | Gross domestic product per capita, current prices | U.S. dollars | Units | See notes for: Gross domestic product, curren... | 354.275 | 2011.0 |
| **New Zealand** | Gross domestic product per capita, current prices | U.S. dollars | Units | See notes for: Gross domestic product, curren... | 37044.891 | 2015.0 |
| **Norway** | Gross domestic product per capita, current prices | U.S. dollars | Units | See notes for: Gross domestic product, curren... | 74822.106 | 2015.0 |

In [102]:

```
plt.figure(figsize=(8,3))

plt.xlabel("GDP per capita")
plt.ylabel('Life satisfaction')

plt.plot(list(sample_data["GDP per capita"]), list(sample_data["Life satisfaction"]), "bo")
plt.plot(list(missing_data["GDP per capita"]), list(missing_data["Life satisfaction"]), "rs

X = np.linspace(0, 110000, 1000)
plt.plot(X, t0full + t1full * X, "r--", label="Linear model on all data")
plt.plot(X, t0 + t1*X, "b:", label="Linear model on partial data")

ridge = linear_model.Ridge(alpha=10**9.5)
Xsample = np.c_[sample_data["GDP per capita"]]
ysample = np.c_[sample_data["Life satisfaction"]]
ridge.fit(Xsample, ysample)
t0ridge, t1ridge = ridge.intercept_[0], ridge.coef_[0][0]
plt.plot(X, t0ridge + t1ridge * X, "b", label="Regularized linear model on partial data")

plt.legend(loc="lower right")
plt.axis([0, 110000, 0, 10])
save_fig('ridge_model_plot')
plt.show()
```

```
Saving figure ridge_model_plot
```




In [103]:

```
backup = oecd_bli, gdp_per_capita

def prepare_country_stats(oecd_bli, gdp_per_capita):
    return sample_data
```

In [104]:

```
# Replace this linear model:
model = sklearn.linear_model.LinearRegression()
```

In [105]:

```
# with this k-neighbors regression model:
model = sklearn.neighbors.KNeighborsRegressor(n_neighbors=3)
```

In [106]:

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

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

# Make a prediction for Cyprus
X_new = np.array([[22587.0]])  # Cyprus' GDP per capita
print(model.predict(X_new)) # outputs [[ 5.76666667]]
```

```
[[5.76666667]]
```

In [ ]:

```

```