Visualizing Data: pandas

There are many ways to interact with data, and one of the most powerful modes of interaction is through **visualizations**. Visualizations show data graphically, and are useful for exploring, analyzing, and presenting datasets. We use four libraries for making visualizations: pandas, Matplotlib, plotly express, and seaborn. In this section, we'll focus on using pandas.

Correlation Matrices

When examining numerical data in columns of a DataFrame, you might want to know how well one column can be approximated as a linear function of another column. In our `mexico-city-real-estate-1` dataset, for example, we might suspect that there was some relationship between the `"price_aprox_usd"` and `"surface_covered_in_m2"` variables. For the sake of thoroughness, let's make a table that shows all the **correlations** in the dataset. the code looks like this:

In [1]:
```
import pandas as pd

columns = ["price_aprox_usd", "surface_covered_in_m2"]
mexico_city1 = pd.read_csv("./data/mexico-city-real-estate-1.csv", usecols=columns)
corr = mexico_city1.corr()
corr.style.background_gradient(axis=None)
```

Out[1]:

	price_aprox_usd	surface_covered_in_m2
price_aprox_usd	1.000000	0.553326
surface_covered_in_m2	0.553326	1.000000

As you can see, there seems to be a moderate, positive correlation between `"price_aprox_usd"` and `"surface_covered_in_m2"` , but there are other relationships here, too. For instance, what if we look at the square root of `"surface_covered_in_m2"` , which is an approximation of a property's length?

In [2]:
```
mexico_city1["length"] = mexico_city1["surface_covered_in_m2"] ** 0.5
corr = mexico_city1.corr()
corr.style.background_gradient(axis=None)
```

Out[2]:

	price_aprox_usd	surface_covered_in_m2	length
price_aprox_usd	1.000000	0.553326	0.607410
surface_covered_in_m2	0.553326	1.000000	0.905212
length	0.607410	0.905212	1.000000

We see that `price_aprox_local_currency` and `price_aprox_usd` have a stronger positive correlation with the `length` of a property than with `surface_covered_in_m2` . This sort of transformation can help improve the performance of a linear model.

Practice

Try it yourself! Repeat the previous calculations for the `mexico-city-real-estate-5.csv` dataset. Is `"length"` better correlated with `"price_aprox_local_currency"` than `"surface_covered_in_m2"` ?

In [3]:
```
# Load CSV into DataFrame
columns = ["price_aprox_local_currency", "surface_covered_in_m2"]
mexico_city5 = pd.read_csv("./data/mexico-city-real-estate-5.csv", usecols=columns)
mexico_city5["length"] = mexico_city5["surface_covered_in_m2"] ** 0.5
corr = mexico_city5.corr()
corr.style.background_gradient(axis=None)
```

Out[3]:

	price_aprox_local_currency	surface_covered_in_m2	length
price_aprox_local_currency	1.000000	-0.004694	0.089524
surface_covered_in_m2	-0.004694	1.000000	0.963806
length	0.089524	0.963806	1.000000

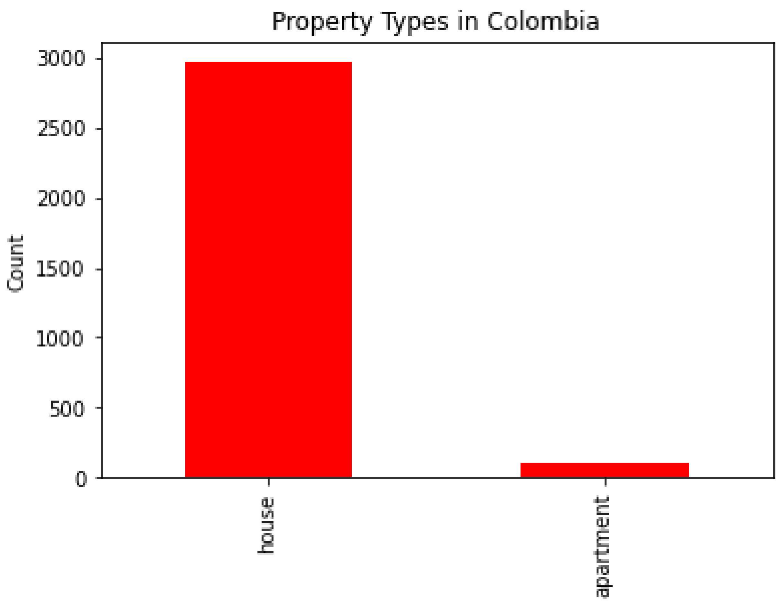
Bar Charts

A **bar chart** is a graph that shows all the values of a categorical variable in a dataset. They consist of an axis and a series of labeled horizontal or vertical bars. The bars depict frequencies of different values of a variable or simply the different values themselves. The numbers on the y-axis of a vertical bar chart or the x-axis of a horizontal bar chart are called the scale.

Let's make a bar chart in pandas using the `colombia-real-estate-1` dataset. We might be curious about how many houses and apartments there are in Colombia, so let's take a look at all the values in the `property_type` variable.

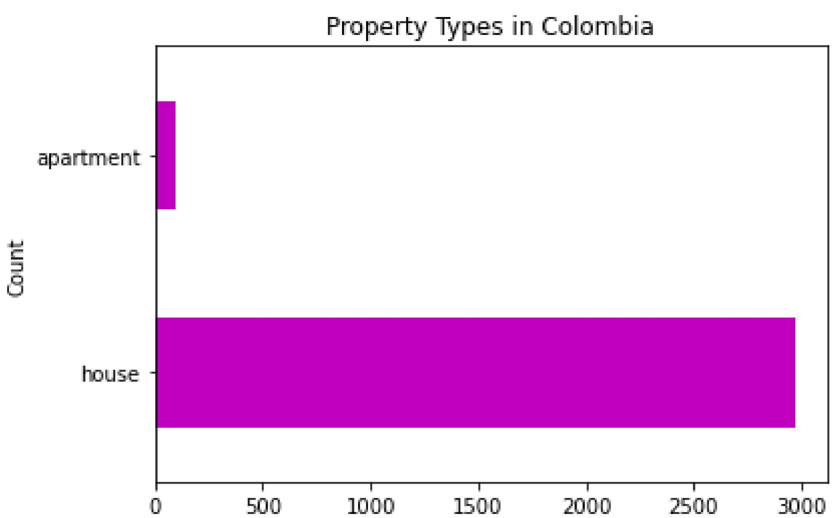
While we often use Matplotlib for our visualizations, pandas has many plotting tools that it borrows from Matplotlib. So we can generate a Series from our DataFrame using `value_counts` and then append the `plot` method to make our visualization. Here's what the code looks like:

In [4]:
```
df1 = pd.read_csv("data/colombia-real-estate-1.csv", usecols=["property_type"])
df1["property_type"].value_counts().plot(
    kind="bar", title="Property Types in Colombia", ylabel="Count", color = 'r'
);
```



If we would prefer a horizontal bar chart (it'll be easier to read the labels), we can change `"bar"` to `"barh"` , like this:

In [5]:
```
df1["property_type"].value_counts().plot(
    kind="barh", title="Property Types in Colombia", ylabel="Count",color = 'm'
);
```



Practice

Try it yourself! Use `value_counts` and the `colombia-real-estate-2` dataset to make a bar chart called `"Property Types in Colombia"` .

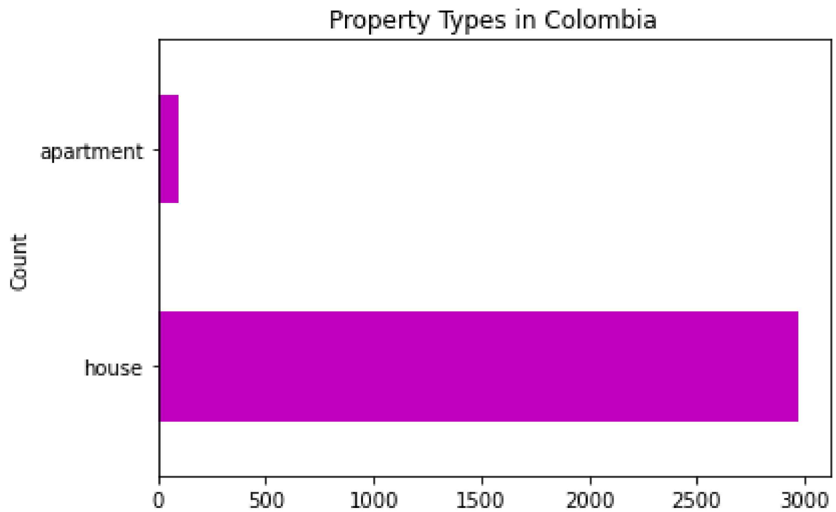
In [6]:
```
df1.columns
```

Out[6]:
```
Index(['property_type'], dtype='object')
```

In [ ]:
```

```

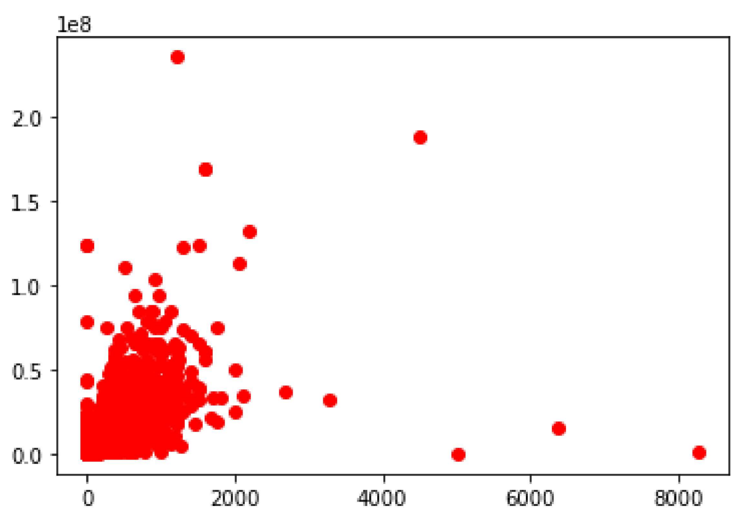
In [7]:
```
df1["property_type"].value_counts().plot(
    kind="barh", title="Property Types in Colombia", ylabel="Count",color = 'm'
);
```



Line Plots

Line plots demonstrate relationships between two variables which have some order. If we look at the data in `mexico-city-real-estate-1.csv` , a scatter plot shows us that there's a relationship between `"surface_covered_in_m2"` and `"price_aprox_local_currency"` .

In [16]:
```
columns = ["surface_covered_in_m2", "price_aprox_local_currency"]
mexico_city1 = pd.read_csv("./data/mexico-city-real-estate-1.csv", usecols=columns)
#mexico_city1.plot.scatter(x="surface_covered_in_m2", y="price_aprox_local_currency");
import matplotlib.pyplot as plt
plt.scatter(x=mexico_city1["surface_covered_in_m2"], y=mexico_city1["price_aprox_local_
```



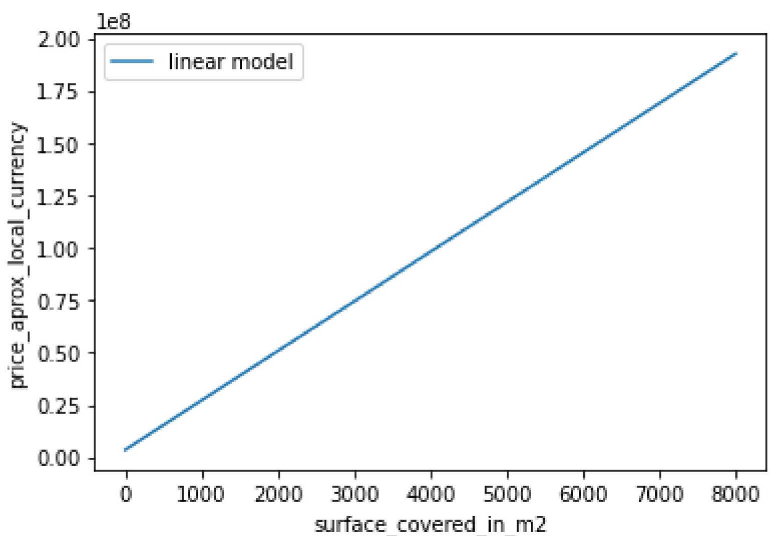
To make clear the relationship between these two features, it would be helpful to have a line showing how price goes up as surface area increases. If we create a linear regression model using this data, the equation for this line would be In Module 2, we determine that the equation for such a line is `price = 3467349 + 23642 * area` . Let's create a series of `x` and `y` values for this line and then plot it.

In [9]:
```
df = pd.DataFrame({"x_coords": range(0, 9000, 1000)})
df["y_coords"] = 3467349 + 23642 * df["x_coords"]
df
```

Out[9]:

	x_coords	y_coords
0	0	3467349
1	1000	27109349
2	2000	50751349
3	3000	74393349
4	4000	98035349
5	5000	121677349
6	6000	145319349
7	7000	168961349
8	8000	192603349

In [10]:
```
df.plot(
    x="x_coords",
    y="y_coords",
    xlabel="surface_covered_in_m2",
    ylabel="price_aprox_local_currency",
    label="linear model",
);
```



Practice

Create a line plot for properties with areas from `0` to `8000` , where the price is determined by the equation `price = 2500000 + 2000 * area` .

In [17]:
```
df = pd.DataFrame({"x_coords": range(0, 9000, 1000)})
df["y_coords"] = 2500000 + 2000 * df["x_coords"]
df
```

Out[17]:

	x_coords	y_coords
0	0	2500000
1	1000	4500000
2	2000	6500000
3	3000	8500000
4	4000	10500000
5	5000	12500000
6	6000	14500000
7	7000	16500000
8	8000	18500000

References & Further Reading

- Online Tutorial on Correlation Matrices using Pandas
- Official Pandas Documentation on Correlations in DataFrames
- Official Pandas Documentation on Styling a Table
- Wikipedia Article on Correlation
- Investopedia Article on Correlation
- Online Tutorial on Correlations
- Pandas Documentation for Bar Charts
- Pandas Official Visualization User Guide
- Pandas Official Documentation on Sorting Values in a DataFrame

---