Toyota Corolla Car Sales

Aayush Dhande

9E

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

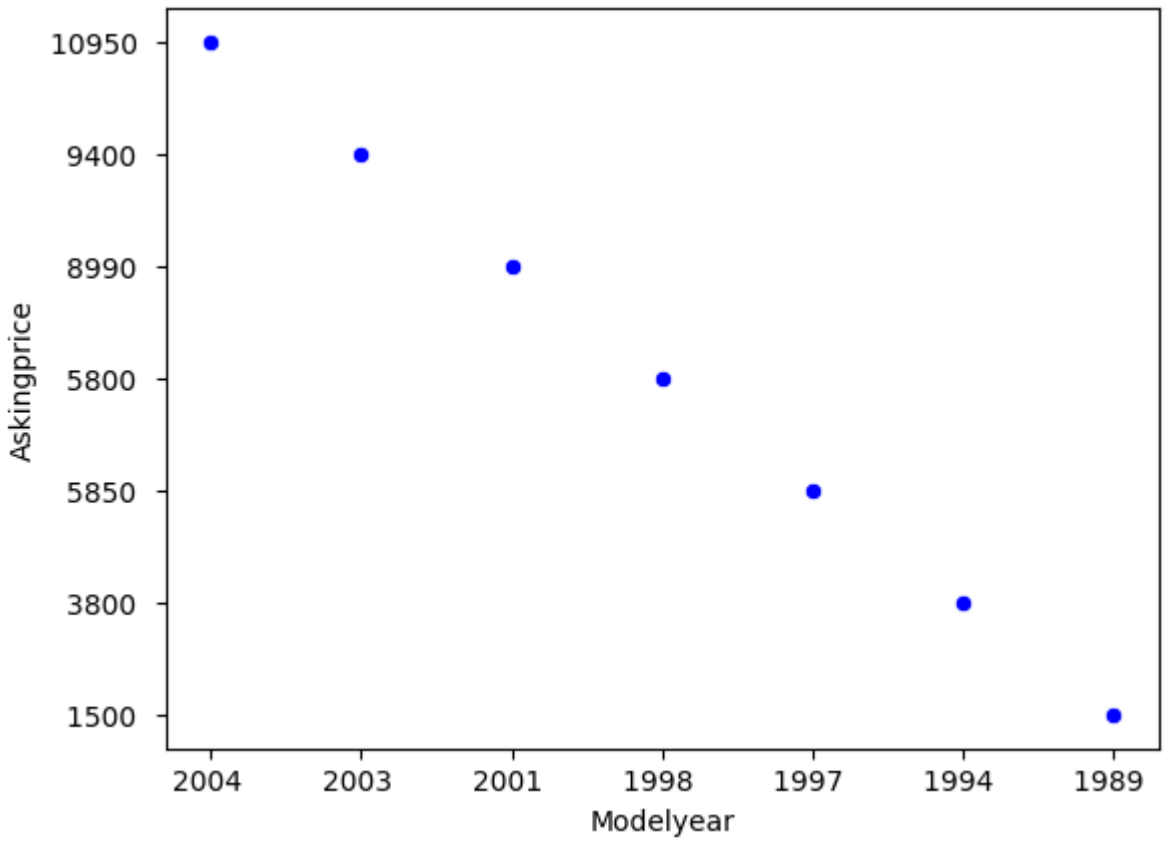
In [ ]:
```
data = pd.read_excel("cardataset1.xlsx")
print(data)
```

```
  Modelyear Askingprice
0      2004       10950
1      2003        9400
2      2001        8990
3      1998        5800
4      1997        5850
5      1994        3800
6      1989        1500
```

Data Visualisation

In [ ]:
```
sns.scatterplot(x="Modelyear",y="Askingprice",data=data,color="blue")
```

Out[ ]:
```
<Axes: xlabel='Modelyear', ylabel='Askingprice'>
```



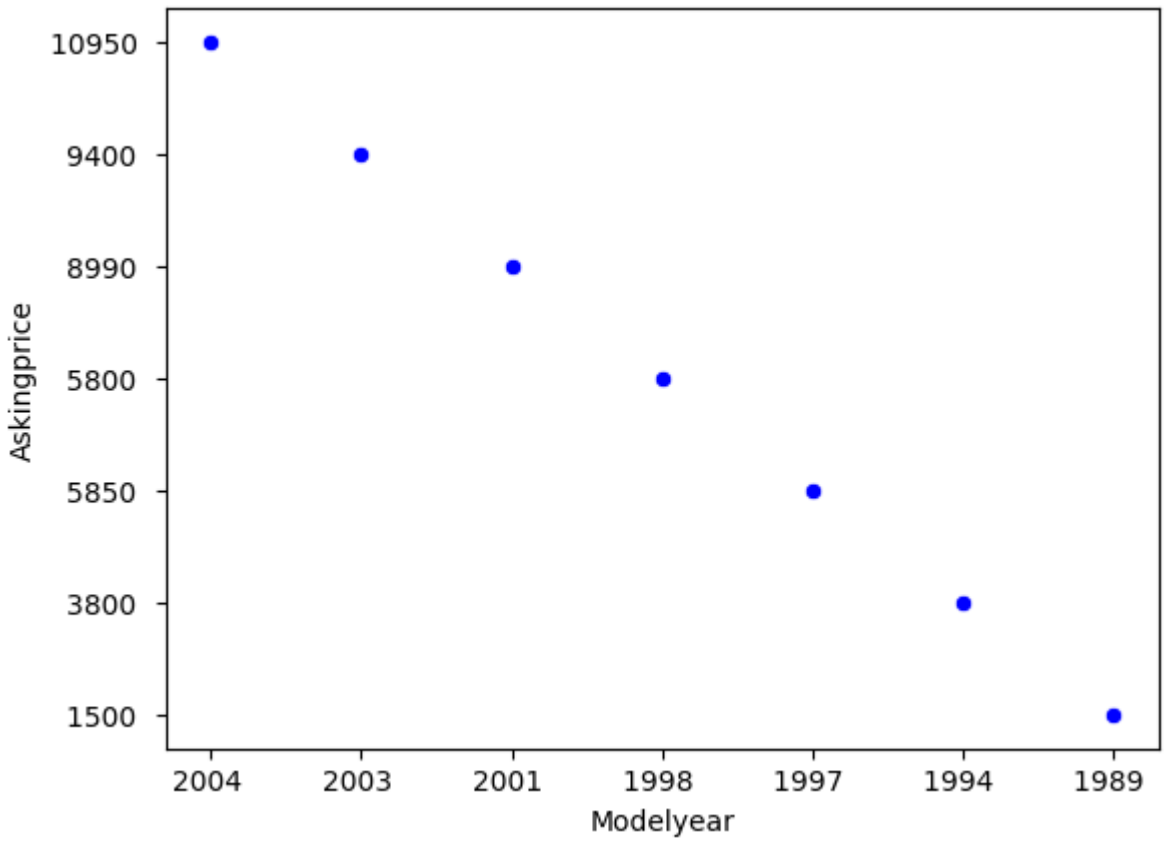
As the data above is in the wrong order

In [ ]:
```
data = pd.read_excel("cardataset1.xlsx")
print(data)
```

```
  Modelyear Askingprice
0      2004       10950
1      2003        9400
2      2001        8990
3      1998        5800
4      1997        5850
5      1994        3800
6      1989        1500
```

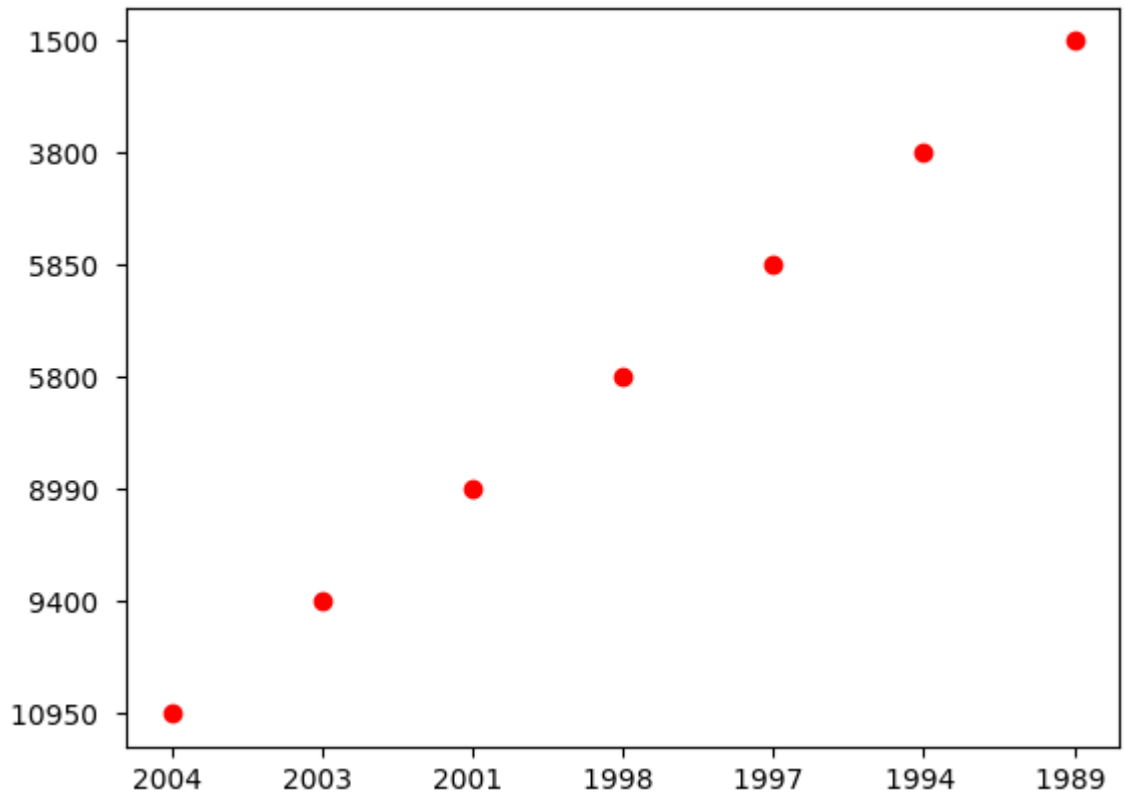
In [ ]:
```
sns.scatterplot(x="Modelyear",y="Askingprice",data=data,color="blue")
```

Out[ ]:
```
<Axes: xlabel='Modelyear', ylabel='Askingprice'>
```

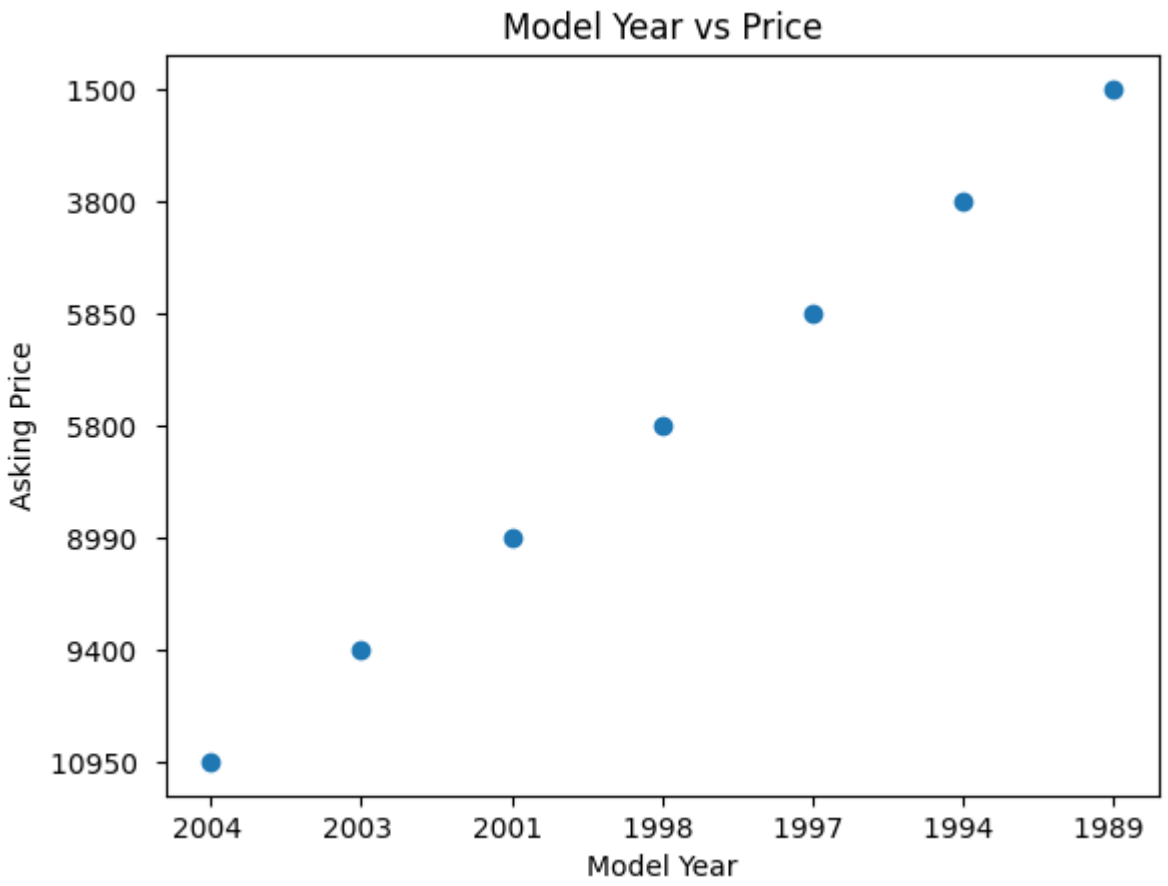


In [ ]:
```
plt.scatter(data.Modelyear,data.Askingprice,color="red")
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x7fe2979b8e80>
```



In [ ]:
```
plt.scatter(data.Modelyear, data.Askingprice)
plt.title("Model Year vs Price")
plt.xlabel( "Model Year")
plt.ylabel( "Asking Price" )
plt.show()
```



In [ ]:
```
x = data.iloc[:,0:1]
y = data.iloc[:,1:]
y.head()
```

Out[ ]:

	Askingprice
0	10950
1	9400
2	8990
3	5800
4	5850

In [ ]:
```
from sklearn.linear_model import LinearRegression
lin = LinearRegression()
lin.fit(x,y)
```

Out[ ]:
```
▾ LinearRegression
LinearRegression()
```

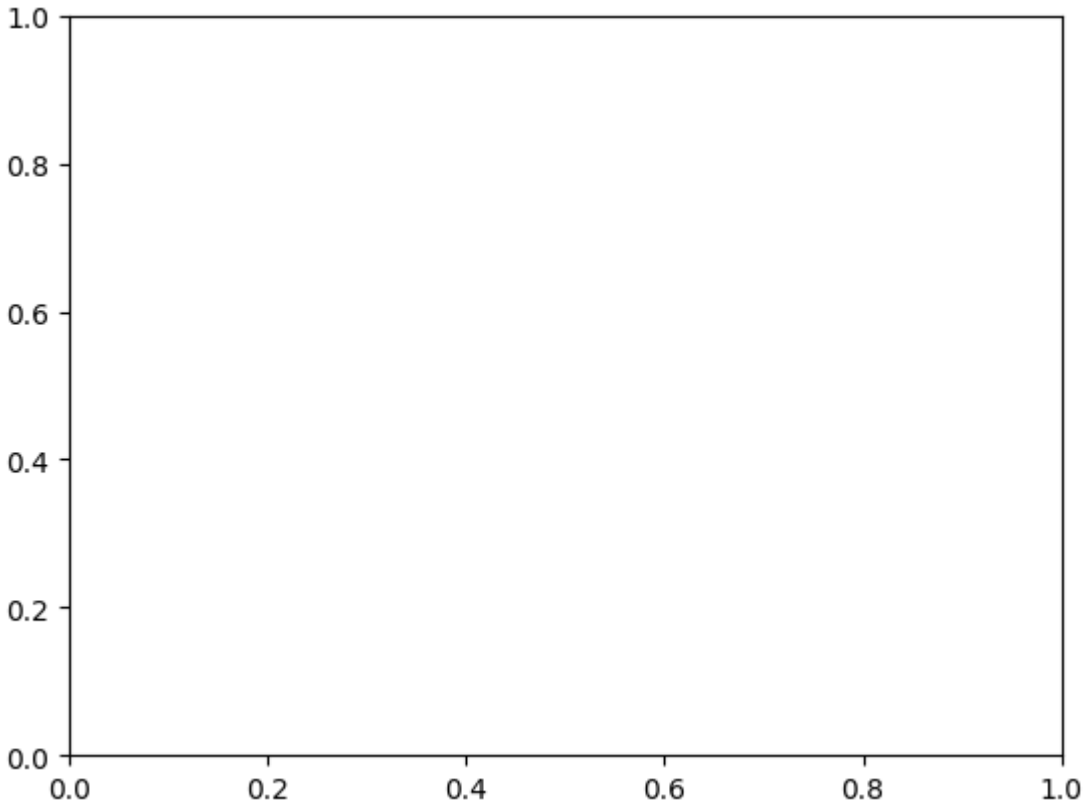
In [ ]:
```
y_prediction = lin.predict(x)
y_prediction
```

Out[ ]:
```
array([[10367.14285714],
       [ 9741.42857143],
       [ 8490.        ],
       [ 6612.85714286],
       [ 5987.14285714],
       [ 4110.        ],
       [  981.42857143]])
```

In [ ]:
```
plt.scatter(x,y)
plt.plot(x,y_prediction,color ="red")
plt.title("Linear Regression")
plt.xlabel("Model Year")
plt.ylabel("Asking Price")
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[14], line 1
----> 1 plt.scatter(x,y)
      2 plt.plot(x,y_prediction,color ="red")
      3 plt.title("Linear Regression")

File ~/.local/lib/python3.10/site-packages/matplotlib/pyplot.py:3684, in scatter(x,
y, s, c, marker, cmap, norm, vmin, vmax, alpha, linewidths, edgecolors, plotnonfinit
e, data, **kwargs)
   3665 @_copy_docstring_and_deprecators(Axes.scatter)
   3666 def scatter(
   3667     x: float | ArrayLike,
   (...)
   3682     **kwargs,
   3683 ) -> PathCollection:
-> 3684     __ret = gca().scatter(
   3685         x,
   3686         y,
   3687         s=s,
   3688         c=c,
   3689         marker=marker,
   3690         cmap=cmap,
   3691         norm=norm,
   3692         vmin=vmin,
   3693         vmax=vmax,
   3694         alpha=alpha,
   3695         linewidths=linewidths,
   3696         edgecolors=edgecolors,
   3697         plotnonfinite=plotnonfinite,
   3698         **({"data": data} if data is not None else {}),
   3699         **kwargs,
   3700     )
   3701     sci(__ret)
   3702     return __ret

File ~/.local/lib/python3.10/site-packages/matplotlib/__init__.py:1465, in _preproce
ss_data.<locals>.inner(ax, data, *args, **kwargs)
   1462 @functools.wraps(func)
   1463 def inner(ax, *args, data=None, **kwargs):
   1464     if data is None:
-> 1465         return func(ax, *map(sanitize_sequence, args), **kwargs)
   1467     bound = new_sig.bind(ax, *args, **kwargs)
   1468     auto_label = (bound.arguments.get(label_namer)
   1469                   or bound.kwargs.get(label_namer))

File ~/.local/lib/python3.10/site-packages/matplotlib/axes/_axes.py:4646, in Axes.sc
atter(self, x, y, s, c, marker, cmap, norm, vmin, vmax, alpha, linewidths, edgecolor
s, plotnonfinite, **kwargs)
   4644 edgecolors = kwargs.pop('edgecolor', None)
   4645 # Process **kwargs to handle aliases, conflicts with explicit kwargs:
-> 4646 x, y = self._process_unit_info([("x", x), ("y", y)], kwargs)
   4647 # np.ma.ravel yields an ndarray, not a masked array,
   4648 # unless its argument is a masked array.
   4649 x = np.ma.ravel(x)
```

```
File ~/.local/lib/python3.10/site-packages/matplotlib/axes/_base.py:2555, in _AxesBase._process_unit_info(self, datasets, kwargs, convert)
   2553     # Update from data if axis is already set but no unit is set yet.
   2554     if axis is not None and data is not None and not axis.have_units():
-> 2555         axis.update_units(data)
   2556 for axis_name, axis in axis_map.items():
   2557     # Return if no axis is set.
   2558     if axis is None:

File ~/.local/lib/python3.10/site-packages/matplotlib/axis.py:1712, in Axis.update_units(self, data)
   1710 neednew = self.converter != converter
   1711 self.converter = converter
-> 1712 default = self.converter.default_units(data, self)
   1713 if default is not None and self.units is None:
   1714     self.set_units(default)

File ~/.local/lib/python3.10/site-packages/matplotlib/category.py:105, in StrCategoryConverter.default_units(data, axis)
    103 # the conversion call stack is default_units -> axis_info -> convert
    104 if axis.units is None:
--> 105     axis.set_units(UnitData(data))
    106 else:
    107     axis.units.update(data)

File ~/.local/lib/python3.10/site-packages/matplotlib/category.py:181, in UnitData.__init__(self, data)
    179 self._counter = itertools.count()
    180 if data is not None:
--> 181     self.update(data)

File ~/.local/lib/python3.10/site-packages/matplotlib/category.py:214, in UnitData.update(self, data)
    212 # check if convertible to number:
    213 convertible = True
--> 214 for val in OrderedDict.fromkeys(data):
    215     # OrderedDict just iterates over unique values in data.
    216     _api.check_isinstance((str, bytes), value=val)
    217     if convertible:
    218         # this will only be called so long as convertible is True.

TypeError: unhashable type: 'numpy.ndarray'
```



I was not able to fix/debug the above error occured in the program in the cell!

We will print the Training Data table here once again to verify the predictions that we will do further

In [ ]:
```
print(data)
```

```
  Modelyear Askingprice
0      2004       10950
1      2003        9400
2      2001        8990
3      1998        5800
4      1997        5850
5      1994        3800
6      1989        1500
```

In [ ]:
```
car_prediction = lin.predict(np.array([[1995]]))
car_prediction
```

```
/home/codespace/.local/lib/python3.10/site-packages/sklearn/base.py:465: UserWarnin
g: X does not have valid feature names, but LinearRegression was fitted with feature
names
  warnings.warn(
```

Out[ ]:
```
array([[4735.71428571]])
```

In [ ]:
```
car_prediction2 = lin.predict(np.array([[2005]]))
car_prediction2
```

```
/home/codespace/.local/lib/python3.10/site-packages/sklearn/base.py:465: UserWarnin
g: X does not have valid feature names, but LinearRegression was fitted with feature
names
  warnings.warn(
```

Out[ ]: array([[10992.85714286]])

Car Prediction 1 (Year : 1995) = $4736~

Car Prediction 2 (Year : 2005) = $10993~

When verified this data with the Training Data, the answer seems to be accurate.

Conclusion : The ML Model is successful