Today Agenda

- Decision Tree Regressor
- Random Forest

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
auto_mobiles_data = pd.read_csv("https://raw.githubusercontent.com/AP-State-Skill-Deve
auto_mobiles_data.head()
```

Out[2]:

	make	fuel-type	num-of-doors	body-style	engine-location	length	width	height	num-of-cylinders	horsepower	pea rp
0	alfa-romero	gas	two	convertible	front	168.8	64.1	48.8	four	111	50(
1	alfa-romero	gas	two	convertible	front	168.8	64.1	48.8	four	111	50(
2	alfa-romero	gas	two	hatchback	front	171.2	65.5	52.4	six	154	50(
3	audi	gas	four	sedan	front	176.6	66.2	54.3	four	102	55(
4	audi	gas	four	sedan	front	176.6	66.4	54.3	five	115	55(

In [3]:

```
# preprocess the data
auto_mobiles_data.isna().sum()
```

Out[3]:

```
make                 0
fuel-type            0
num-of-doors         0
body-style           0
engine-location      0
length               0
width                0
height               0
num-of-cylinders     0
horsepower           0
peak-rpm             0
city-mpg             0
highway-mpg          0
price                0
dtype: int64
```

In [4]:

```python
auto_mobiles_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 14 columns):
make                201 non-null object
fuel-type           201 non-null object
num-of-doors        201 non-null object
body-style          201 non-null object
engine-location     201 non-null object
length              201 non-null float64
width               201 non-null float64
height              201 non-null float64
num-of-cylinders    201 non-null object
horsepower          201 non-null object
peak-rpm            201 non-null object
city-mpg            201 non-null int64
highway-mpg         201 non-null int64
price               201 non-null int64
dtypes: float64(3), int64(3), object(8)
memory usage: 22.1+ KB
```

In [5]:

```python
from sklearn.preprocessing import LabelEncoder
lab = LabelEncoder()
```

In [6]:

```python
auto_mobiles_data['make'].unique()
```

Out[6]:

```
array(['alfa-romero', 'audi', 'bmw', 'chevrolet', 'dodge', 'honda',
       'isuzu', 'jaguar', 'mazda', 'mercedes-benz', 'mercury',
       'mitsubishi', 'nissan', 'peugot', 'plymouth', 'porsche', 'renault',
       'saab', 'subaru', 'toyota', 'volkswagen', 'volvo'], dtype=object)
```

In [7]:

```python
auto_mobiles_data['make'] = lab.fit_transform(
auto_mobiles_data['make'])
auto_mobiles_data.head()
```

Out[7]:

	make	fuel-type	num-of-doors	body-style	engine-location	length	width	height	num-of-cylinders	horsepower	peak-rpm
0	0	gas	two	convertible	front	168.8	64.1	48.8	four	111	5000
1	0	gas	two	convertible	front	168.8	64.1	48.8	four	111	5000
2	0	gas	two	hatchback	front	171.2	65.5	52.4	six	154	5000
3	1	gas	four	sedan	front	176.6	66.2	54.3	four	102	5500
4	1	gas	four	sedan	front	176.6	66.4	54.3	five	115	5500

In [8]:

```
auto_mobiles_data['fuel-type'] = lab.fit_transform(
    auto_mobiles_data['fuel-type'])
auto_mobiles_data['fuel-type'].head()
```

Out[8]:

```
0    1
1    1
2    1
3    1
4    1
Name: fuel-type, dtype: int32
```

In [11]:

```
dum = lab.fit_transform(auto_mobiles_data['num-of-doors'])
auto_mobiles_data['num-of-doors']=dum
```

In [17]:

```
#'body-style','engine-location'
auto_mobiles_data['body-style'] = lab.fit_transform(
    auto_mobiles_data['body-style'])
```

In [18]:

```
auto_mobiles_data.head()
```

```
...
```

In [19]:

```
auto_mobiles_data['engine-location'] = lab.fit_transform(
auto_mobiles_data['engine-location'])
```

In [20]:

```
auto_mobiles_data['num-of-cylinders'] = lab.fit_transform(
auto_mobiles_data['num-of-cylinders'])
```

In [21]:

```
auto_mobiles_data.info()
```

```
...
```

In [22]:

```
auto_mobiles_data['horsepower'] = lab.fit_transform(
auto_mobiles_data['horsepower'])
```

In [23]:

```
auto_mobiles_data['peak-rpm'] = lab.fit_transform(
auto_mobiles_data['peak-rpm'])
```

In [24]:

```
auto_mobiles_data.info()
```

...

In [26]:

```
dup = auto_mobiles_data.corr()
dup
```

Out[26]:

	make	fuel-type	num-of-doors	body-style	engine-location	length	width	heig
make	1.000000	-0.109330	-0.121393	0.090621	0.053312	0.110468	-0.005115	0.2307
fuel-type	-0.109330	1.000000	0.206001	-0.147654	0.040917	-0.211187	-0.244356	-0.2815
num-of-doors	-0.121393	0.206001	1.000000	-0.672697	0.139671	-0.370590	-0.212729	-0.5135
body-style	0.090621	-0.147654	-0.672697	1.000000	-0.278350	0.347571	0.155366	0.5711
engine-location	0.053312	0.040917	0.139671	-0.278350	1.000000	-0.053086	-0.052205	-0.1092
length	0.110468	-0.211187	-0.370590	0.347571	-0.053086	1.000000	0.857170	0.4920
width	-0.005115	-0.244356	-0.212729	0.155366	-0.052205	0.857170	1.000000	0.3060
height	0.230754	-0.281578	-0.513564	0.571107	-0.109225	0.492063	0.306002	1.0000
num-of-cylinders	-0.049947	0.120638	0.182172	-0.063741	0.136009	-0.111660	-0.158449	-0.3170
horsepower	0.102312	-0.086710	-0.135986	0.135149	-0.000673	-0.223649	-0.236179	0.1485
peak-rpm	-0.202725	0.492531	0.210074	-0.091028	0.182240	-0.274166	-0.227662	-0.3142
city-mpg	0.065761	-0.265676	0.012041	0.014217	-0.157132	-0.665192	-0.633531	-0.0498
highway-mpg	0.059111	-0.198690	0.029000	-0.021328	-0.102964	-0.698142	-0.680635	-0.1048
price	-0.163646	-0.110326	-0.032289	-0.072933	0.331062	0.690628	0.751265	0.1354

In [27]:

```
dup['price']
```

Out[27]:

```
make                -0.163646
fuel-type           -0.110326
num-of-doors        -0.032289
body-style          -0.072933
engine-location      0.331062
length               0.690628
width                0.751265
height               0.135486
num-of-cylinders     0.005509
horsepower          -0.333078
peak-rpm            -0.111070
city-mpg            -0.686571
highway-mpg         -0.704692
price                1.000000
Name: price, dtype: float64
```

In [28]:

```
auto_mobiles_data.columns
```

Out[28]:

```
Index(['make', 'fuel-type', 'num-of-doors', 'body-style', 'engine-location',
       'length', 'width', 'height', 'num-of-cylinders', 'horsepower',
       'peak-rpm', 'city-mpg', 'highway-mpg', 'price'],
      dtype='object')
```

In [29]:

```
input_data = auto_mobiles_data[['make', 'fuel-type', 'num-of-doors', 'body-style', 'eng
       'length', 'width', 'height', 'num-of-cylinders', 'horsepower',
       'peak-rpm', 'city-mpg', 'highway-mpg']]
```

In [30]:

```
output_data = auto_mobiles_data['price']
```

In [31]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(input_data,
                                                 output_data,
                                                 random_state=1,
                                                 test_size=0.3)
```

In [32]:

```
from sklearn.tree import DecisionTreeRegressor
dtr = DecisionTreeRegressor()
```

In [33]:

```
# train the model
dtr.fit(x_train,y_train)
```

Out[33]:

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

In [34]:

```
dtr.score(x_train,y_train)
```

Out[34]:

```
0.9952864478639523
```

In [35]:

```
pred = dtr.predict(x_test)
```

In [36]:

```
dtr.score(x_test,y_test)
```

Out[36]:

```
0.7389171731775055
```

```
Tasks:
- Take corr > 0 work on this Algorithm and compare with this accuracy
- For this dataset check the pairplot
- Print the tree in notebbok for above regressor example(Decision Tree Regressor)
```

Random Forest

```
- It is the ensemble model
- Ensemble: Group of models
- It is also working on gini and entropy
```

In [38]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [39]:

```
data = pd.read_csv("https://raw.githubusercontent.com/AP-State-Skill-Development-Corpo
data.head()
```

Out[39]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28

In [40]:

```
data.columns
```

Out[40]:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insuli
n',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [41]:

```
data.isna().sum()
```

...

In [42]:

```
data.info()
```

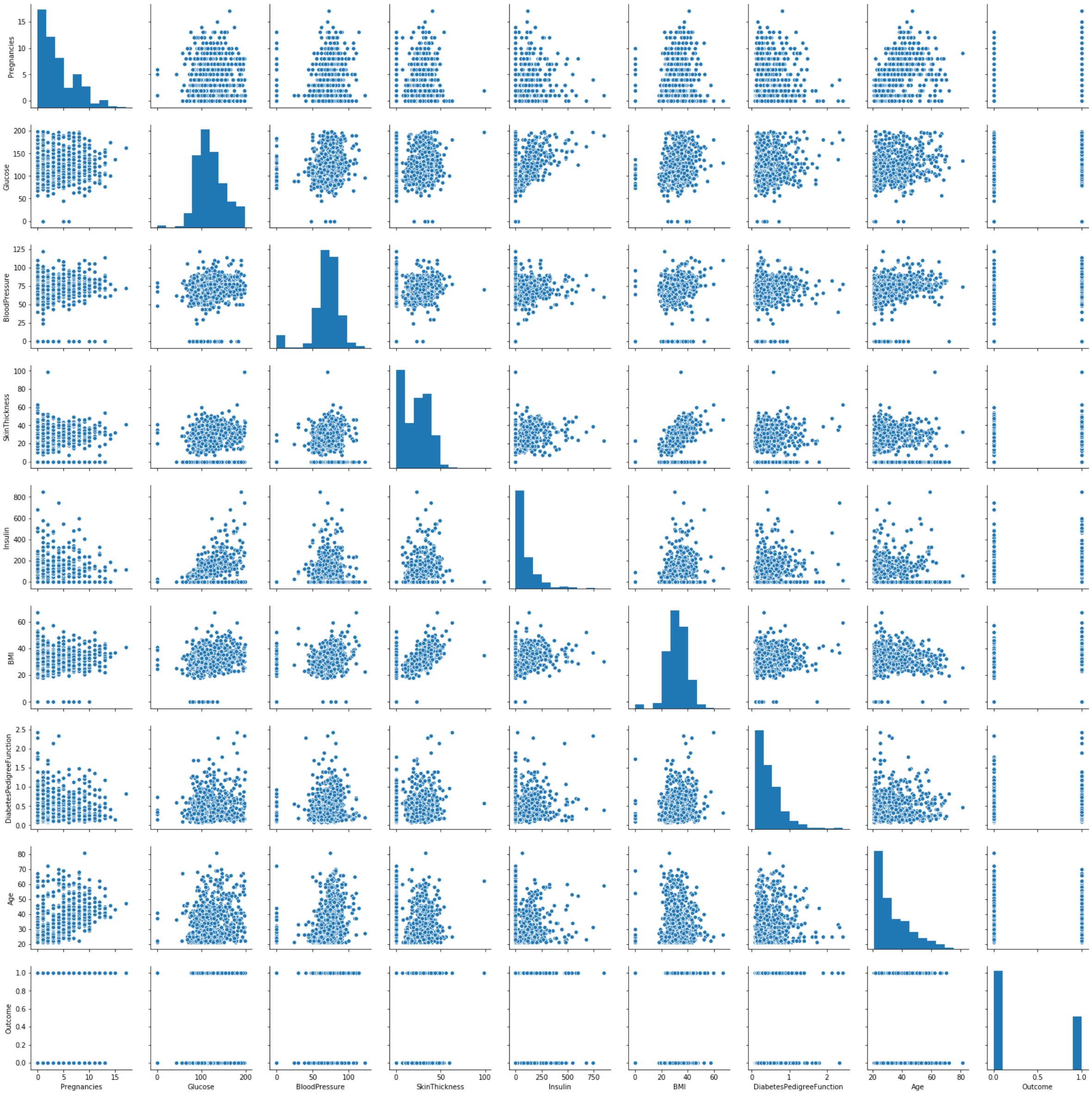
...

In [43]:

```
sns.pairplot(data)
```

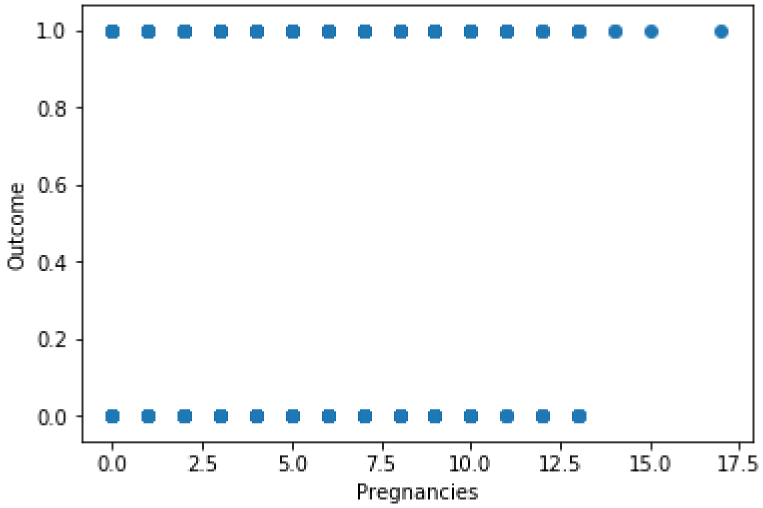
Out[43]:

```
<seaborn.axisgrid.PairGrid at 0x1a4ec3ab780>
```



In [47]:

```
plt.scatter(data['Pregnancies'],data['Outcome'])
plt.xlabel("Pregnancies")
plt.ylabel("Outcome")
plt.show()
```



In [48]:

```
data_input = data.drop('Outcome',axis=1)
```

In [49]:

```
data_output=data['Outcome']
```

In [50]:

```
data.shape
```

Out[50]:

```
(768, 9)
```

In [51]:

```
data_input.shape
```

Out[51]:

```
(768, 8)
```

In [52]:

```
data_output.shape
```

Out[52]:

```
(768,)
```

In [53]:

```
from sklearn.model_selection import train_test_split
x_tr,x_te,y_tr,y_te = train_test_split(data_input,data_output,
                                       random_state=0,
                                      test_size=0.4)
```

In [54]:

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
```

In [56]:

```
rfc.fit(x_tr,y_tr)
```

Out[56]:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

In [57]:

```
pred_val = rfc.predict(x_te)
```

In [58]:

```
from sklearn.metrics import accuracy_score,confusion_matrix
from sklearn.metrics import classification_report
```

In [59]:

```
accuracy_score(y_te,pred_val)
```

Out[59]:

```
0.7532467532467533
```

In [60]:

```
confusion_matrix(y_te,pred_val)
```

Out[60]:

```
array([[179,  26],
       [ 50,  53]], dtype=int64)
```

In [62]:

```
print(classification_report(y_te,pred_val))
```

```
              precision    recall  f1-score   support

           0       0.78      0.87      0.82       205
           1       0.67      0.51      0.58       103

   micro avg       0.75      0.75      0.75       308
   macro avg       0.73      0.69      0.70       308
weighted avg       0.74      0.75      0.74       308
```

Random Forest Regressor

In [ ]:

```
import pandas as pd
```

In [ ]:

```
house_data = pd.read_csv("https://raw.githubusercontent.com/AP-State-Skill-Development
house_data.head()
```

In [ ]:

```

```