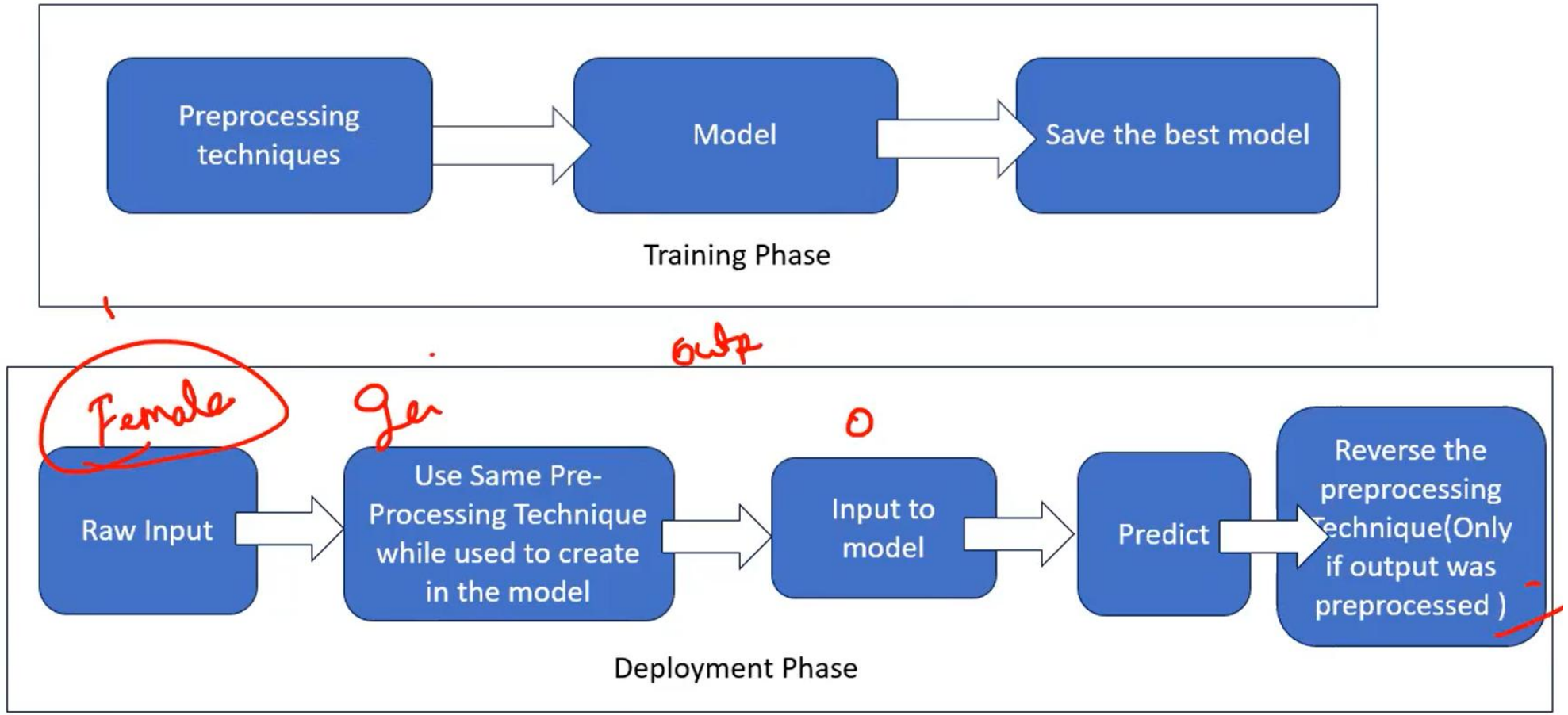
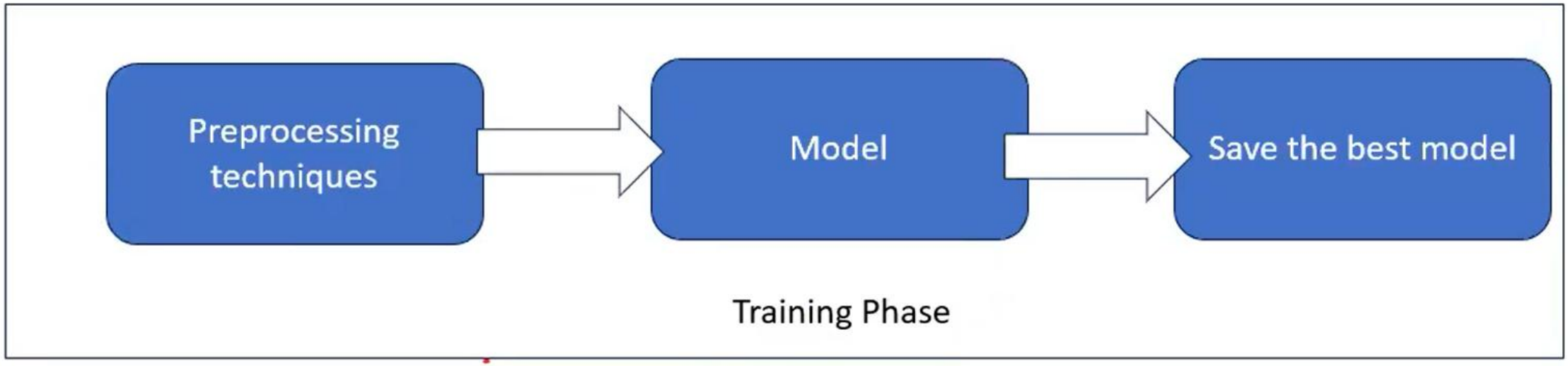
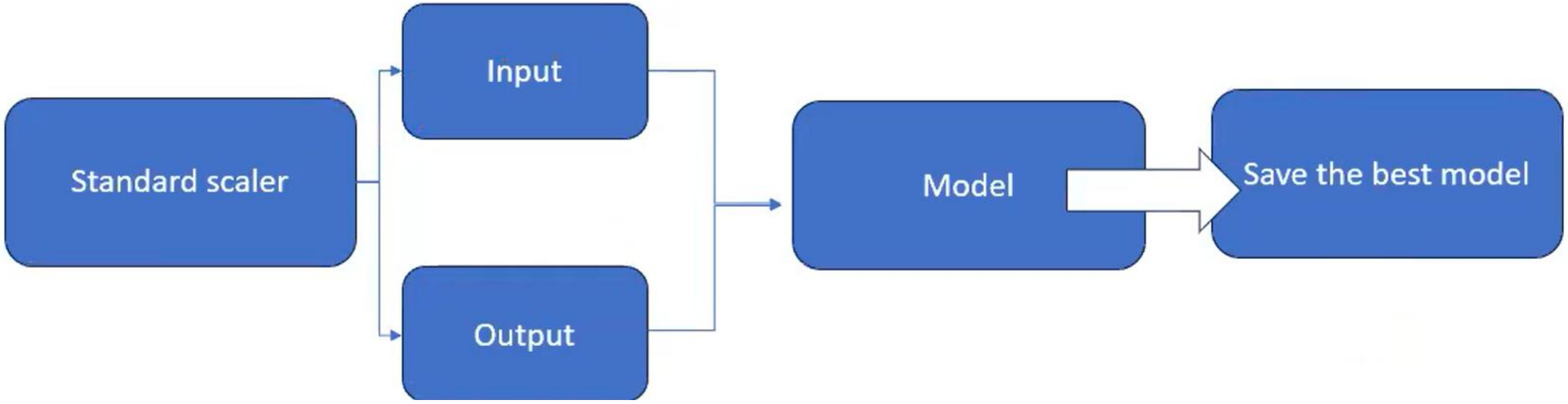
AdvancedTechqniue-
How to handle preprocessed Input and Output during Deployment



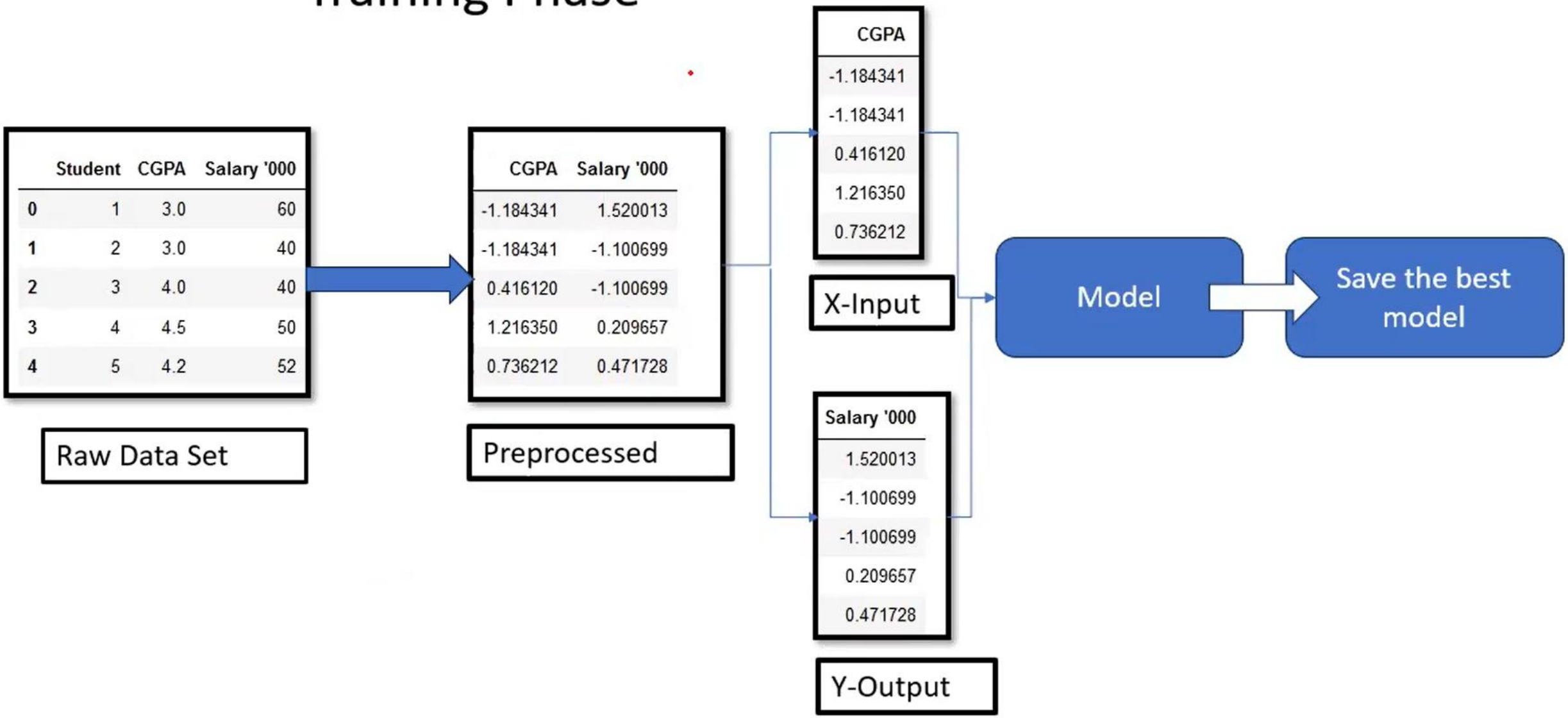
Advanced Technique- Case-1- PrePro X and Y

Training Phase

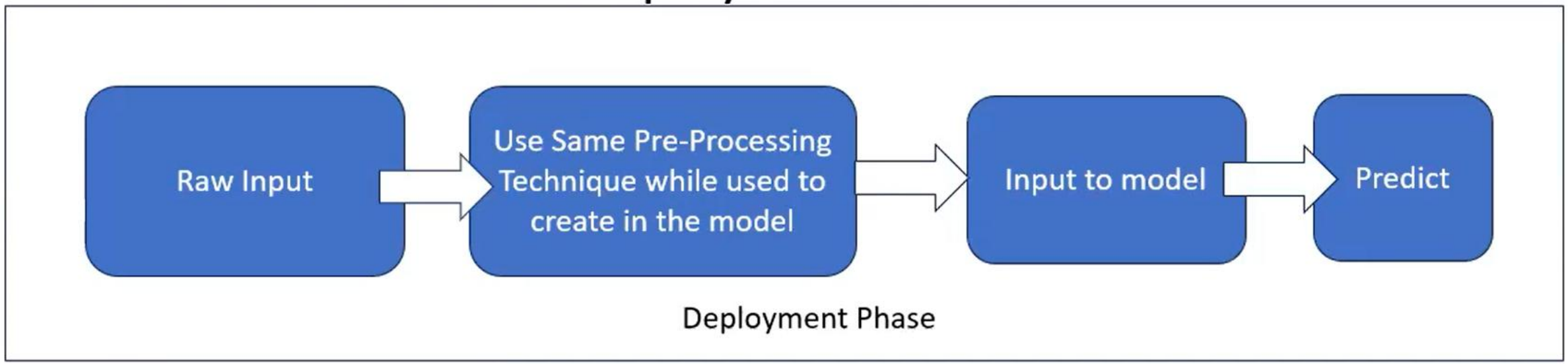


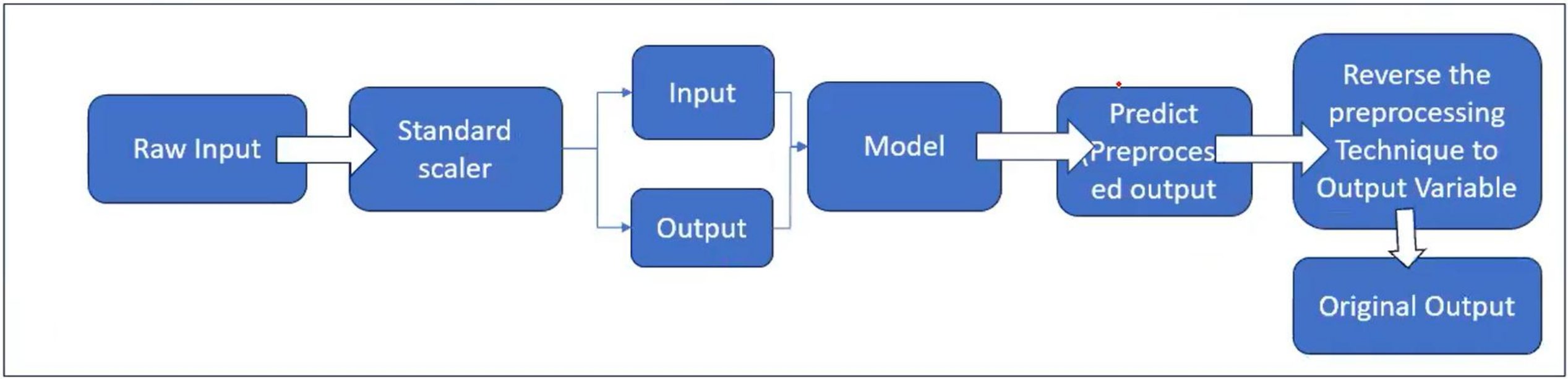


Training Phase

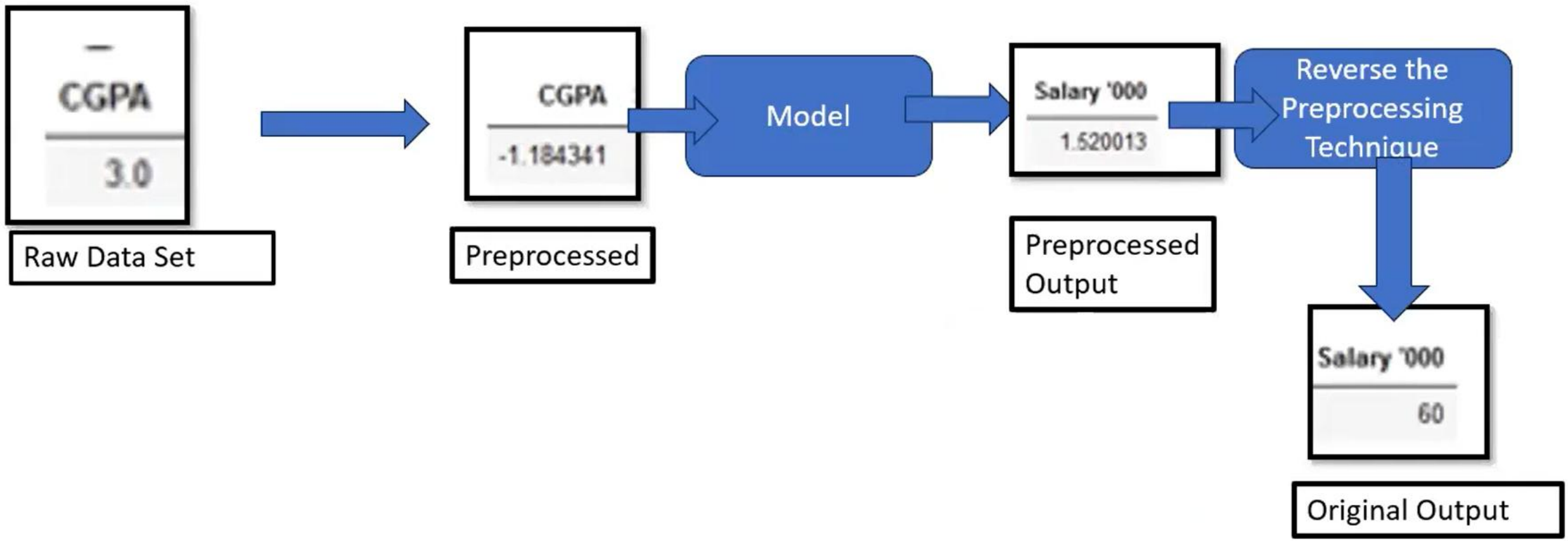


Deployment Phase

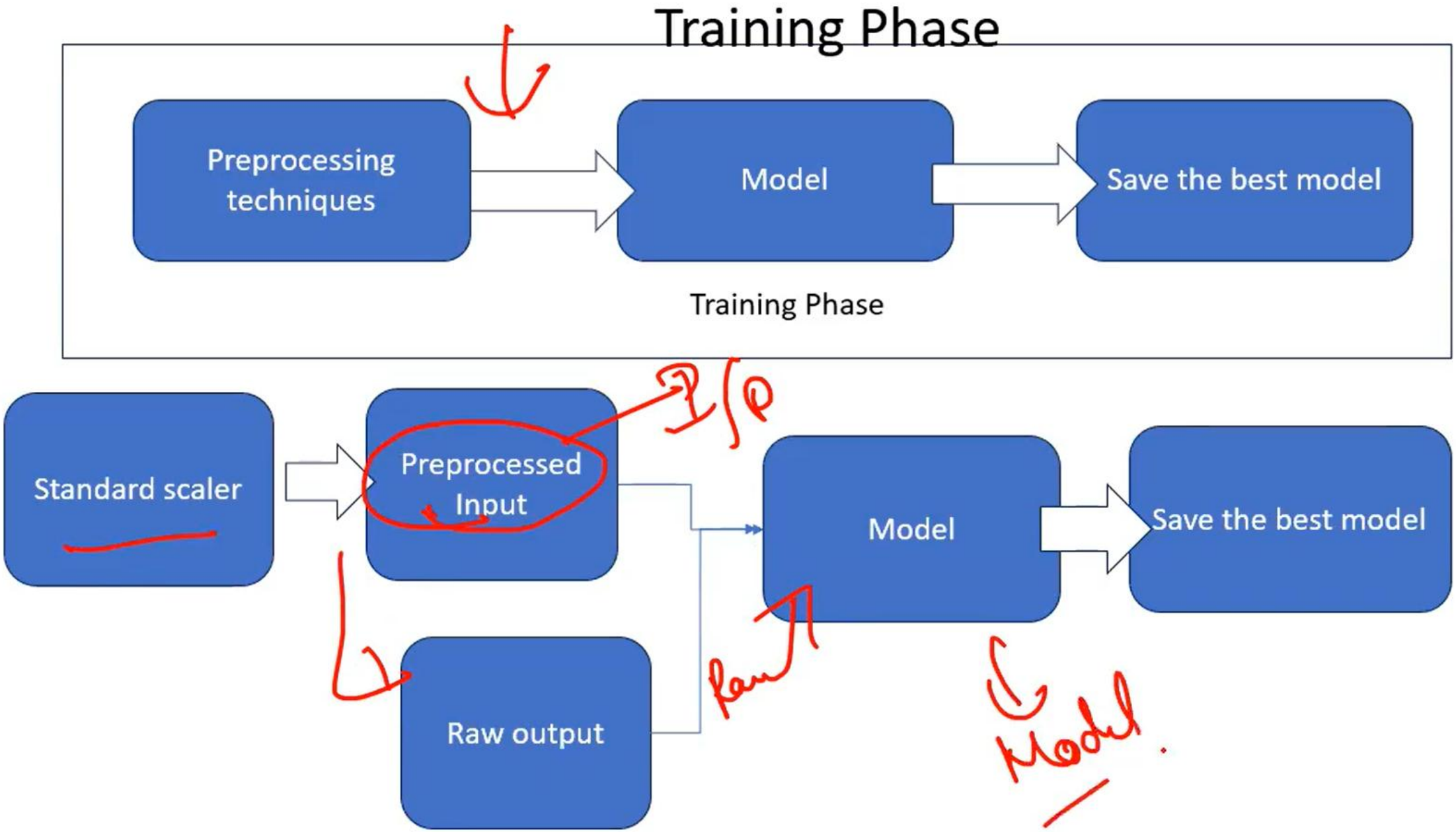




Deployment Phase



Advanced Technique- Case-2 –Prepro X only



Advanced Technique- Case-2 –Prepro X

Training Phase

	Student	CGPA	Salary '000
0	1	3.0	60
1	2	3.0	40
2	3	4.0	40
3	4	4.5	50
4	5	4.2	52

Raw Data Set

CGPA
-1.184341
-1.184341
0.416120
1.216350
0.736212

Preprocessed

Preprocessed X-Input

CGPA
-1.184341
-1.184341
0.416120
1.216350
0.736212

Salary '000
60
40
40
50
52

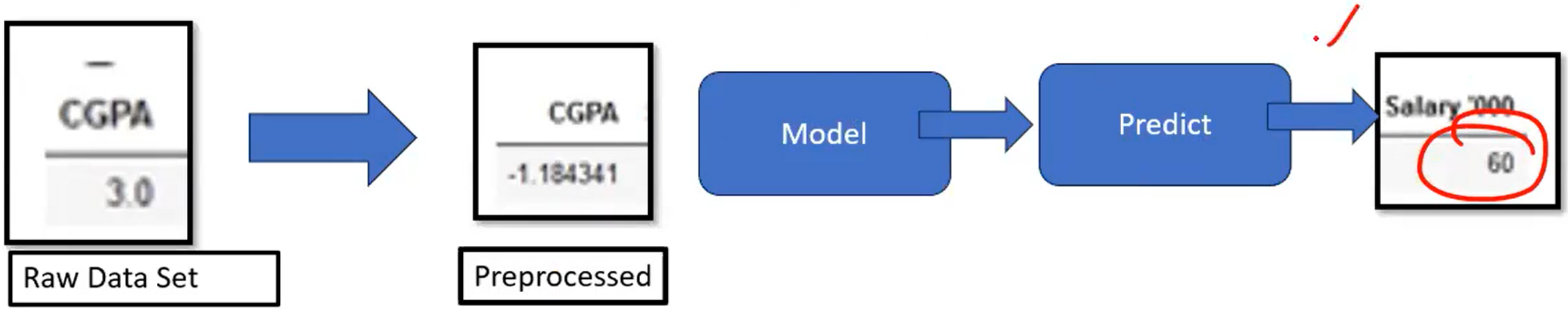
Raw values Y-output

Model

Save the best model

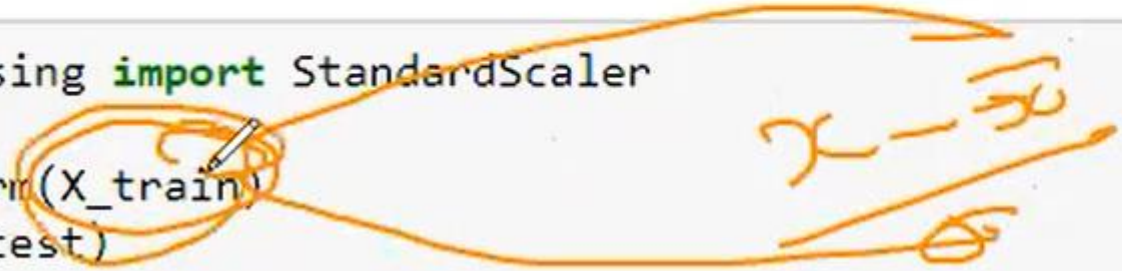
Advanced Technique- Case-2 –Prepro X

Deployment Phase



3	144372.41	118671.85	383199.62	0	1
0	165349.20	136897.80	471784.10	0	1
47	0.00	135426.92	0.00	0	0
44	22177.74	154806.14	28334.72	0	0

In [11]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```



In [12]:
```
X_train
```

Out[12]:
```
array([[ 1.17644103,  0.84515251,  0.94354978,  2.        , -0.76870611],
       [ 0.96420324,  1.27283565,  0.42738817,  2.        , -0.76870611],
       [-1.47369826,  0.0153175 , -1.52350329, -0.5       ,  1.30088727],
       [-1.48308929, -2.79556363, -1.53809178, -0.5       ,  1.30088727],
       [-0.14952431,  1.13637282, -0.71716495, -0.5       ,  1.30088727],
       [ 0.85312042, -0.04431628,  0.46771725, -0.5       ,  1.30088727],
       [-0.22353674, -0.3151007 , -0.83981652,  2.        , -0.76870611],
       [-0.19454707,  0.21199679, -1.18497259, -0.5       , -0.76870611],
       [ 0.10478723, -0.08388412,  0.48740807, -0.5       , -0.76870611],
       [-1.0096458 , -1.07019473, -0.4040623 , -0.5       , -0.76870611],
       [ 0.06872897, -0.38396487,  0.75036616, -0.5       , -0.76870611],
```

44	22177.74	154806.14	28334.72	0	0

In [12]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

In [13]:
```
scy=StandardScaler()
y_train=scy.fit_transform(y_train)
y_test=scy.transform(y_test)
```

In [14]:
```
X_train
```

Out[14]:
```
array([[ 1.17644103,  0.84515251,  0.94354978,  2.        , -0.76870611],
       [ 0.96420324,  1.27283565,  0.42738817,  2.        , -0.76870611],
       [-1.47369826,  0.0153175 , -1.52350329, -0.5       ,  1.30088727],
       [-1.48308929, -2.79556363, -1.53809178, -0.5       ,  1.30088727],
       [ 0.14952431,  1.13637282,  0.71716495, -0.5       ,  1.30088727]
```

In [20]:
```
import pickle
filename="finalized_model_svr.sav"
```

In [21]:
```
pickle.dump(regressor,open(filename,'wb'))
```

In [24]:
```
preinput=sc.transform([[1300,12000,4000,0,1]])
```

In [25]:
```
preinput
```

Out[25]:
```
array([[-1.46755405, -4.33835385, -1.50744257, -0.5       ,  1.30088727]])
```

In [26]:
```
loaded_model=pickle.load(open("finalized_model_svr.sav",'rb'))
result=loaded_model.predict(preinput)
```

In [27]:
```
result
```

Out[27]:
```
array([107679.41510409])
```

In [21]:
```
import pickle
filename="finalized_model_svr.sav"
```

In [22]:
```
pickle.dump(regressor,open(filename,'wb'))
```

In [23]:
```
preinput=sc.transform([[1300,12000,4000,0,1]])
```

In [25]:
```
preinput
```

Out[25]:
```
array([[-1.46755405, -4.33835385, -1.50744257, -0.5       ,  1.30088727]])
```

In [26]:
```
loaded_model=pickle.load(open("finalized_model_svr.sav",'rb'))
result=loaded_model.predict(preinput)
```

In [27]:
```
result
```

Out[27]:
```
array([-1.03109284])
```

In [32]:
```
preoutput=scy.inverse_transform([result])
```

In [33]:
```
preoutput
```

Out[33]:
```
array([[65875.16964031]])
```

Methods

`fit`(X[, y, sample_weight])	Compute the mean and std to be used for later scaling.
`fit_transform`(X[, y])	Fit to data, then transform it.
`get_feature_names_out`([input_features])	Get output feature names for transformation.
`get_metadata_routing`()	Get metadata routing of this object.
`get_params`([deep])	Get parameters for this estimator.
`inverse_transform`(X[, copy])	Scale back the data to the original representation.
`partial_fit`(X[, y, sample_weight])	Online computation of mean and std on X for later scaling.
`set_fit_request`(*[, sample_weight])	Request metadata passed to the `fit` method.
`set_inverse_transform_request`(*[, copy])	Request metadata passed to the `inverse_transform` method.
`set_output`(*[, transform])	Set output container.
`set_params`(**params)	Set the parameters of this estimator.
`set_partial_fit_request`(*[, sample_weight])	Request metadata passed to the `partial_fit` method.
`set_transform_request`(*[, copy])	Request metadata passed to the `transform` method.
`transform`(X[, copy])	Perform standardization by centering and scaling.

In [1]:
```
import pandas as pd
dataset=pd.read_csv("50_Startups.csv")
dataset=pd.get_dummies(dataset,drop_first=True)
independent=dataset[['R&D Spend', 'Administration', 'Marketing Spend','State_Florida', 'State_New York']]
dependent=dataset[["Profit"]]
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(independent, dependent, test_size=0.30,random_state=0)
```

In [2]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

In [3]:
```
scy=StandardScaler()
y_train=scy.fit_transform(y_train)
y_test=scy.transform(y_test)
```

In [4]:
```
preinput=sc.transform([[1300,12000,4000,0,1]])
```

In [5]:
```
preinput
```

Out[5]:
```
array([[-1.46755405, -4.33835385, -1.50744257, -0.5       ,  1.30088727]])
```

In [7]:
```
import pickle
loaded_model=pickle.load(open("finalized_model_svr.sav",'rb'))
result=loaded_model.predict(preinput)
```

```
C:\Anaconda3\envs\aiml\lib\site-packages\sklearn\linear_model\least_angle.py:30: DeprecationWarning: `np.float` is a
```