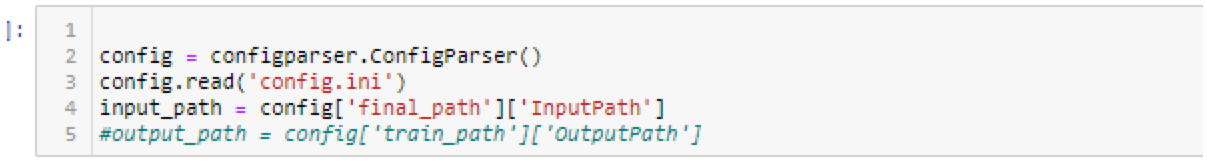
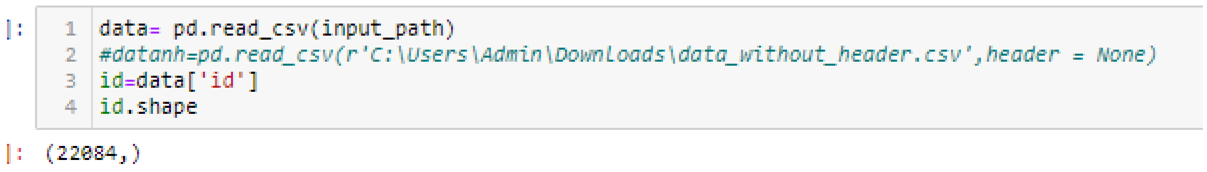
Load Data by CSV file



```

config = configparser.ConfigParser()
config.read('config.ini')
input_path = config['final_path']['InputPath']
#output_path = config['train_path']['OutputPath']
```



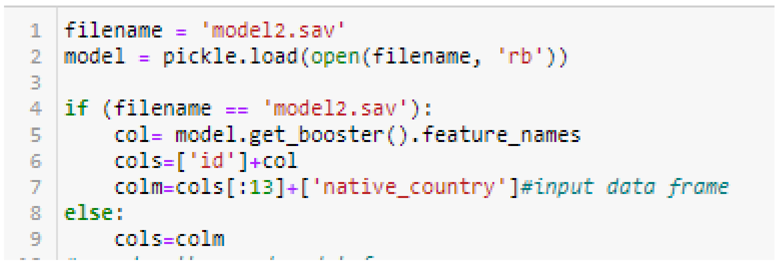
```
data= pd.read_csv(input_path)
#datanh=pd.read_csv(r'C:\Users\Admin\Downloads\data_without_header.csv',header = None)
id=data['id']
id.shape
```

```
(22084,)
```

Model loading

We have use pickle to load model

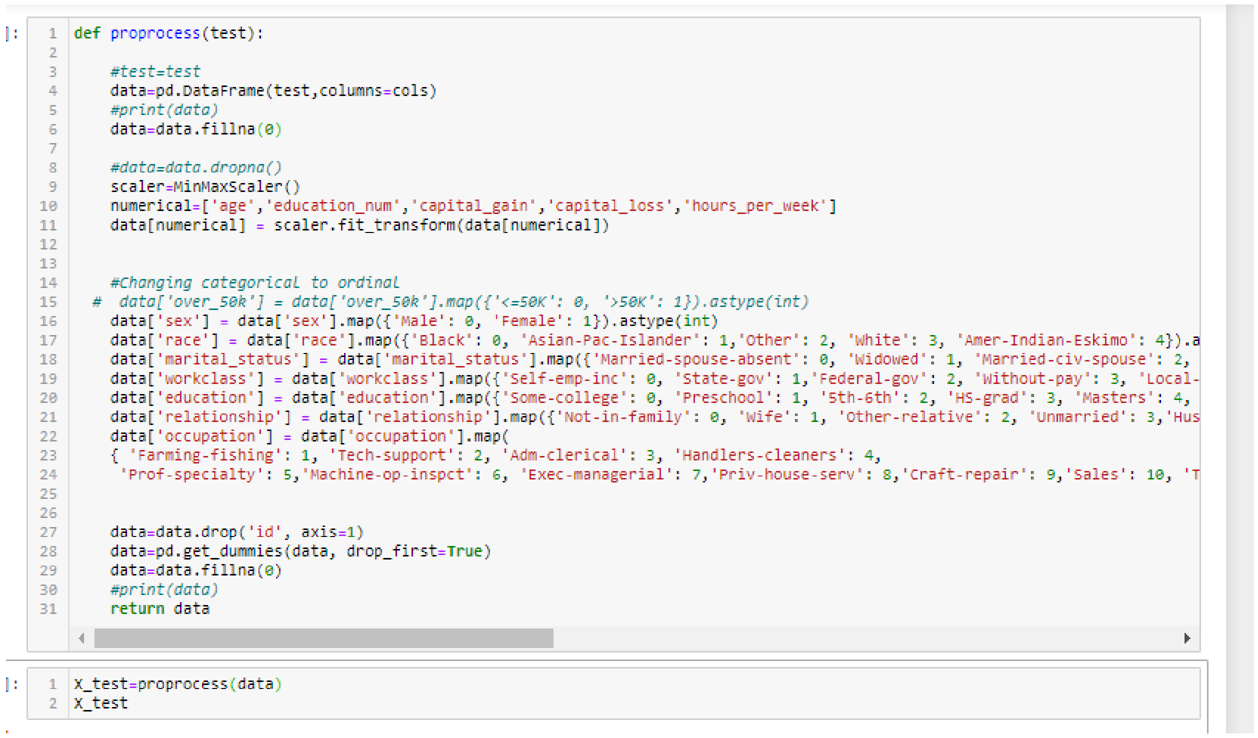
For using xgboost model we need to change order in column order



```
filename = 'model2.sav'
model = pickle.load(open(filename, 'rb'))

if (filename == 'model2.sav'):
    col= model.get_booster().feature_names
    cols=['id']+col
    colm=cols[:13]+['native_country']#input data frame
else:
    cols=colm
```

Function for pre processing

we need to process the data according to the model input

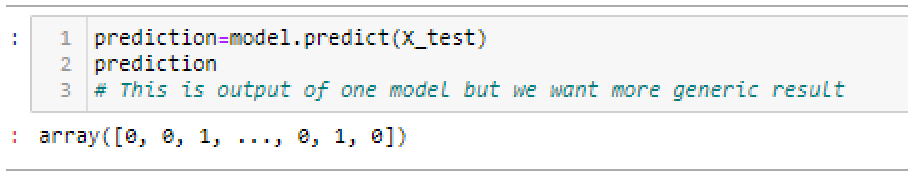


```
]:  def proprocess(test):

        #test=test
        data=pd.DataFrame(test,columns=cols)
        #print(data)
        data=data.fillna(0)

        #data=data.dropna()
        scaler=MinMaxScaler()
        numerical=['age','education_num','capital_gain','capital_loss','hours_per_week']
        data[numerical] = scaler.fit_transform(data[numerical])


        #Changing categorical to ordinal
      #  data['over_50k'] = data['over_50k'].map({'<=50K': 0, '>50K': 1}).astype(int)
        data['sex'] = data['sex'].map({'Male': 0, 'Female': 1}).astype(int)
        data['race'] = data['race'].map({'Black': 0, 'Asian-Pac-Islander': 1,'Other': 2, 'White': 3, 'Amer-Indian-Eskimo': 4}).a
        data['marital_status'] = data['marital_status'].map({'Married-spouse-absent': 0, 'Widowed': 1, 'Married-civ-spouse': 2,
        data['workclass'] = data['workclass'].map({'Self-emp-inc': 0, 'State-gov': 1,'Federal-gov': 2, 'Without-pay': 3, 'Local-
        data['education'] = data['education'].map({'Some-college': 0, 'Preschool': 1, '5th-6th': 2, 'HS-grad': 3, 'Masters': 4,
        data['relationship'] = data['relationship'].map({'Not-in-family': 0, 'Wife': 1, 'Other-relative': 2, 'Unmarried': 3,'Hus
        data['occupation'] = data['occupation'].map(
        { 'Farming-fishing': 1, 'Tech-support': 2, 'Adm-clerical': 3, 'Handlers-cleaners': 4,
         'Prof-specialty': 5,'Machine-op-inspct': 6, 'Exec-managerial': 7,'Priv-house-serv': 8,'Craft-repair': 9,'Sales': 10, 'T


        data=data.drop('id', axis=1)
        data=pd.get_dummies(data, drop_first=True)
        data=data.fillna(0)
        #print(data)
        return data
```

```
]:  X_test=proprocess(data)
    X_test
```

Single Model prediction

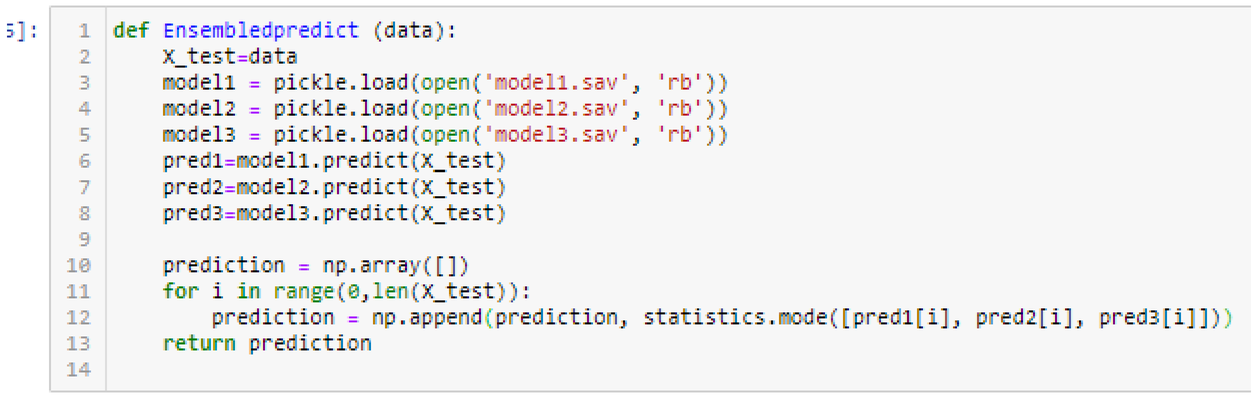
Prediction from single model.if you directly run same prediction after this block it will save this data



```
:  prediction=model.predict(X_test)
   prediction
   # This is output of one model but we want more generic result

:  array([0, 0, 1, ..., 0, 1, 0])
```

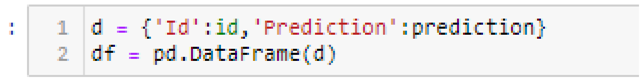
Ensembling Model

Ensemble of ANN,Random Forest and Xgboost

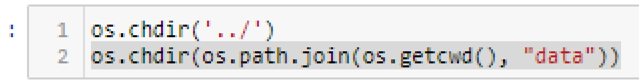


```python
def Ensembledpredict (data):
    X_test=data
    model1 = pickle.load(open('model1.sav', 'rb'))
    model2 = pickle.load(open('model2.sav', 'rb'))
    model3 = pickle.load(open('model3.sav', 'rb'))
    pred1=model1.predict(X_test)
    pred2=model2.predict(X_test)
    pred3=model3.predict(X_test)

    prediction = np.array([])
    for i in range(0,len(X_test)):
        prediction = np.append(prediction, statistics.mode([pred1[i], pred2[i], pred3[i]]))
    return prediction

```

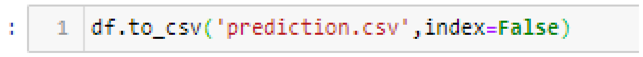
Prediction



```python
d = {'Id':id,'Prediction':prediction}
df = pd.DataFrame(d)
```

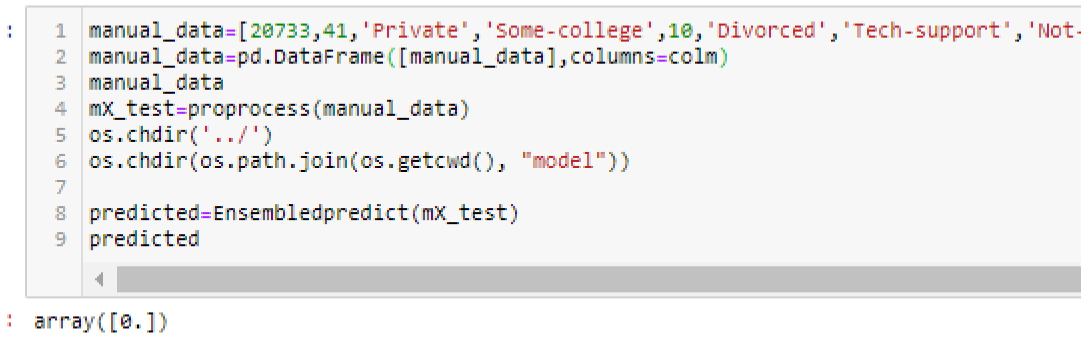


```python
os.chdir('../')
os.chdir(os.path.join(os.getcwd(), "data"))
```



```python
df.to_csv('prediction.csv',index=False)
```

Prediction for a single manual data



```python
manual_data=[20733,41,'Private','Some-college',10,'Divorced','Tech-support','Not-
manual_data=pd.DataFrame([manual_data],columns=colm)
manual_data
mX_test=proprocess(manual_data)
os.chdir('../')
os.chdir(os.path.join(os.getcwd(), "model"))

predicted=Ensembledpredict(mX_test)
predicted
```

```
array([0.])
```