Salary Prediction Application

End to end application for salary prediction
using Gradient Boosting and Flask

By: Vaibhav Mehta

Case Study

The aim here is to develop a machine learning model for the prediction of salary range of the people who want to opt for a loan based on which a banking organisation can decide if a person will be able to repay the loan and predict the maximum amount of loan that could be approved for a new applier. The model has been trained here using the Gradient Boosting Classifier and uses the past dataset provided for the training and testing of the model. The application has been developed here using the Flask framework of python.

Data Features

- Independent Features:
    - Age
    - Workclass
    - Fnlwgt
    - Education - Level of education attained
    - Educational-num - No of years of education
    - Marital-status
    - Occupation
    - Relationship
    - Race
    - Gender
    - Capital-gain
    - Capital-loss
    - Hours-per-week - The number of hours a user works
    - Native-country - Country of origin

Data Features Contd.

- Dependent Variable:
    - Income - If the income is less than 50k or more than that

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

Gradient Boosting Algorithm

Gradient boosting is one of the most powerful techniques for building predictive models. Gradient boosting is a machine learning technique for regression, classification and other tasks, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient boosted trees, which usually outperforms random forest. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function (y=ax+b+e , e needs a special mention as it is the error term). The loss function is a measure indicating how good are model's coefficients are at fitting the underlying data. A logical understanding of loss function would depend on what we are trying to optimise.

```
sklearn.ensemble.GradientBoostingClassifier(name_of_model,           loss='deviance',
learning_rate=0.1,    n_estimators=100,     subsample=1.0,     criterion='friedman_mse',
min_samples_split=2,  min_samples_leaf=1,  min_weight_fraction_leaf=0.0,   max_depth=3,
min_impurity_decrease=0.0,    init=None,      random_state=None,      max_features=None,
verbose=0,      max_leaf_nodes=None,      warm_start=False,      validation_fraction=0.1,
n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0)
```

Pickle Module

The pickle module implements binary protocols for serializing and de-serializing a Python object structure. *“Pickling”* is the process whereby a Python object hierarchy is converted into a byte stream, and *“unpickling”* is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as “serialization”, “marshalling,” 1 or “flattening”.

import pickle ----> Used for importing the pickle module in python

pickle.**dump**(obj, file, protocol=None, *, fix_imports=True, buffer_callback=None) ----->Write the pickled representation of the object obj to the open file object file.

pickle.**load**(file, *, fix_imports=True, encoding="ASCII", errors="strict", buffers=None) -----> Read the pickled representation of an object from the open file object file and return the reconstituted object hierarchy specified therein.

ML - Code Screenshots

```
import pandas
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import numpy
df = pandas.read_csv("adult.csv")
col_names = df.columns
for c in col_names:
    df[c] = df[c].replace("?", numpy.NaN)
df = df.apply(lambda x:x.fillna(x.value_counts().index[0]))
df.replace(['Divorced', 'Married-AF-spouse','Married-civ-spouse', 'Married-spouse-absent','Never-married','Separated','Widowed'],
             ['divorced','married','married','married','not married','not married','not married'], inplace = True)
category_columns =['workclass', 'race', 'education','marital-status', 'occupation','relationship', 'gender', 'native-country',
'income']
labelEncoder = preprocessing.LabelEncoder()
mapping_dict={}
for col in category_columns:
    df[col] = labelEncoder.fit_transform(df[col])
    le_name_mapping = dict(zip(labelEncoder.classes_, labelEncoder.transform(labelEncoder.classes_)))
    mapping_dict[col]=le_name_mapping
df=df.drop(['fnlwgt','educational-num'], axis=1)
```

```python
X = df.values[:, 0:12]
Y = df.values[:,12]
X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size = 0.3, random_state = 100)

from sklearn import ensemble
from sklearn.ensemble import GradientBoostingClassifier
boost=GradientBoostingClassifier(loss='deviance',
                                 learning_rate=0.1,
                                 n_estimators=200,
                                 min_samples_leaf=5,
                                 max_depth=5,
                                 verbose=1)
boost.fit(X_train,y_train)
y_pred=boost.predict(X_test)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
print(cm)
print ("Gradient Boosting Classifier with 200 estimators and learning rate as 0.1 \nAccuracy is) "
from sklearn.metrics import f1_score
print(f1_score(y_test, y_pred, average='micro'))

import pickle
pickle.dump(boost, open("model.pkl","wb"))
```

ML - Code Outputs

```
      Iter       Train Loss   Remaining Time
         1           1.0298            6.21s
         2           0.9735            6.11s
         3           0.9283            6.11s
         4           0.8914            6.06s
         5           0.8589            6.02s
         6           0.8322            5.99s
         7           0.8089            6.04s
         8           0.7881            5.99s
         9           0.7706            5.98s
        10           0.7531            5.98s
        20           0.6564            5.81s
        30           0.6109            5.51s
        40           0.5857            5.19s
        50           0.5683            4.86s
        60           0.5556            4.53s
        70           0.5460            4.20s
        80           0.5382            3.89s
        90           0.5325            3.57s
       100           0.5264            3.24s
       200           0.4887            0.00s
[[6991  437]
 [ 783 1558]]
Gradient Boosting Classifier with 200 estimators and learning rate as 0.1
Accuracy is
0.8751151602006346
```

Flask Code

```python
import os
import numpy as np
import flask
import pickle
from flask import Flask, render_template, request

app=Flask(__name__)
@app.route('/')
@app.route('/index')

def index():
    return flask.render_template('index.html')

def ValuePredictor(to_predict_list):
    to_predict = np.array(to_predict_list).reshape(1,12)
    loaded_model = pickle.load(open("model.pkl","rb"))
    result = loaded_model.predict(to_predict)
    return result[0]
```

Flask Code contd.

```
@app.route('/result',methods = ['POST'])

def result():
    if request.method == 'POST':
        to_predict_list = request.form.to_dict()
        to_predict_list=list(to_predict_list.values())
        to_predict_list = list(map(int, to_predict_list))
        result = ValuePredictor(to_predict_list)
        if int(result)==1:
            prediction='Allow them to have a loan their income is more than 50K.'
        else:
            prediction='Have a caution as probably their income is less than 50K.'

        return render_template("result.html",prediction=prediction)

if __name__ == "__main__":
  app.run(debug=True)
```

Front-end Code for index.html

```
<html>
<head>
    <style>
    label{
        display: inline-block;
        width:200px;
        text-align: right;
        font-size: 20px;
        color:green;
        font-style
        }
    input[type=text] {
        width: 250px;
        padding: 10px 10px;
        margin: 5px 20px;
        border: 2px solid black;
        border-radius: 4px;
        background-color: aqua;
    }
    select {
        width: 250px;
        padding: 10px 20px;
        margin: 5px 20px;
        border: 2px solid black;
        border-radius: 4px;
        background-color: aqua;
        }
    input[type=text]:focus {
        border: 3px solid #555;
    }
```

```
    select:focus {
        border: 3px solid #555;
    }
    input[type=submit]{
        text-align:center;
        background-color: green;
        border: none;
        color: white;
        padding: 16px 32px;
        text-decoration: none;
        margin: auto;
        cursor: pointer;
    }
    div{
        text-align:center;
        width:500px;
        border-radius:5px;
        padding: 20px;
        background-image: url('myimg.jpg');
        background-size: 700px;
        margin: auto;
    }
    body{
        background-image: radial-gradient(aqua, yellow, aqua);
    }
    h3{
        text-align:center;
        font-size:40px;
        color:blue;
    }
    </style>
</head>
```

```html
<body>
    <h3><b><u>Income Prediction Form</u></b></h3>
<div>
  <form action="/result" method="POST">
    <label for="age">Age</label>
    <input type="text" id="age" name="age" placeholder="Enter valid age"><br>
    <br>
    <label for="w_class">Working Class</label>
    <select id="w_class" name="w_class">
      <option value="0">Federal-gov</option>
      <option value="1">Local-gov</option>
      <option value="2">Never-worked</option>
      <option value="3">Private</option>
      <option value="4">Self-emp-inc</option>
      <option value="5">Self-emp-not-inc</option>
      <option value="6">State-gov</option>
      <option value="7">Without-pay</option>
    </select>
    <br><br>
    <label for="edu">Education</label>
    <select id="edu" name="edu">
      <option value="0">10th</option>
      <option value="1">11th</option>
      <option value="2">12th</option>
      <option value="3">1st-4th</option>
      <option value="4">5th-6th</option>
      <option value="5">7th-8th</option>
      <option value="6">9th</option>
      <option value="7">Assoc-acdm</option>
      <option value="8">Assoc-voc</option>
      <option value="9">Bachelors</option>
      <option value="10">Doctorate</option>
      <option value="11">HS-grad</option>
      <option value="12">Masters</option>
      <option value="13">Preschool</option>
      <option value="14">Prof-school</option>
      <option value="15">16 - Some-college</option>
    </select>
    <br><br>
```

```html
<label for="martial_stat">Marital Status</label>
<select id="martial_stat" name="martial_stat">
  <option value="0">divorced</option>
  <option value="1">married</option>
  <option value="2">not married</option>
</select>
<br><br>
<label for="occup">Occupation</label>
<select id="occup" name="occup">
  <option value="0">Adm-clerical</option>
  <option value="1">Armed-Forces</option>
  <option value="2">Craft-repair</option>
  <option value="3">Exec-managerial</option>
  <option value="4">Farming-fishing</option>
  <option value="5">Handlers-cleaners</option>
  <option value="6">Machine-op-inspct</option>
  <option value="7">Other-service</option>
  <option value="8">Priv-house-serv</option>
  <option value="9">Prof-specialty</option>
  <option value="10">Protective-serv</option>
  <option value="11">Sales</option>
  <option value="12">Tech-support</option>
  <option value="13">Transport-moving</option>
</select>
<br><br>
<label for="relation">Relationship</label>
<select id="relation" name="relation">
  <option value="0">Husband</option>
  <option value="1">Not-in-family</option>
  <option value="2">Other-relative</option>
  <option value="3">Own-child</option>
  <option value="4">Unmarried</option>
  <option value="5">Wife</option>
</select>
<br><br>
```

```
<label for="race">Race</label>
<select id="race" name="race">
  <option value="0">Amer Indian Eskimo</option>
  <option value="1">Asian Pac Islander</option>
  <option value="2">Black</option>
  <option value="3">Other</option>
  <option value="4">White</option>
</select>
<br><br>
<label for="gender">Gender</label>
<select id="gender" name="gender">
  <option value="0">Female</option>
  <option value="1">Male</option>
</select>
<br><br>
<label for="c_gain">Capital Gain </label>
<input type="text" id="c_gain" name="c_gain" placeholder="Between 0
    -99999">
<br><br>
<label for="c_loss">Capital Loss </label>
<input type="text" id="c_loss" name="c_loss" placeholder="Between 0
    -4356">
<br><br>
<label for="hours_per_week">Hours per Week </label>
<input type="text" id="hours_per_week" name="hours_per_week"
    placeholder="Between 1-99">
<br><br>
<label for="native-country">Native Country</label>
<select id="native-country" name="native-country">
  <option value="0">Cambodia</option>
  <option value="1">Canada</option>
  <option value="2">China</option>
  <option value="3">Columbia</option>
  <option value="4">Cuba</option>
  <option value="5">Dominican Republic</option>
  <option value="6">Ecuador</option>
  <option value="7">El Salvadorr</option>
  <option value="8">England</option>
  <option value="9">France</option>
  <option value="10">Germany</option>
```

```
      <option value="7">El Salvadorr</option>
      <option value="8">England</option>
      <option value="9">France</option>
      <option value="10">Germany</option>
      <option value="11">Greece</option>
      <option value="12">Guatemala</option>
      <option value="13">Haiti</option>
      <option value="14">Netherlands</option>
      <option value="15">Honduras</option>
      <option value="16">HongKong</option>
      <option value="17">Hungary</option>
      <option value="18">India</option>
      <option value="19">Iran</option>
      <option value="20">Ireland</option>
      <option value="21">Italy</option>
      <option value="22">Jamaica</option>
      <option value="23">Japan</option>
      <option value="24">Laos</option>
      <option value="25">Mexico</option>
      <option value="26">Nicaragua</option>
      <option value="27">Outlying-US(Guam-USVI-etc)</option>
      <option value="28">Peru</option>
      <option value="29">Philippines</option>
      <option value="30">Poland</option>
      <option value="11">Portugal</option>
      <option value="32">Puerto-Rico</option>
      <option value="33">Scotland</option>
      <option value="34">South</option>
      <option value="35">Taiwan</option>
      <option value="36">Thailand</option>
      <option value="37">Trinadad&Tobago</option>
      <option value="38">United States</option>
      <option value="39">Vietnam</option>
      <option value="40">Yugoslavia</option>
    </select>
    <br><br><br>
    <input type="submit" value="Submit">
  </form>
</div>
</body>
</html>
```

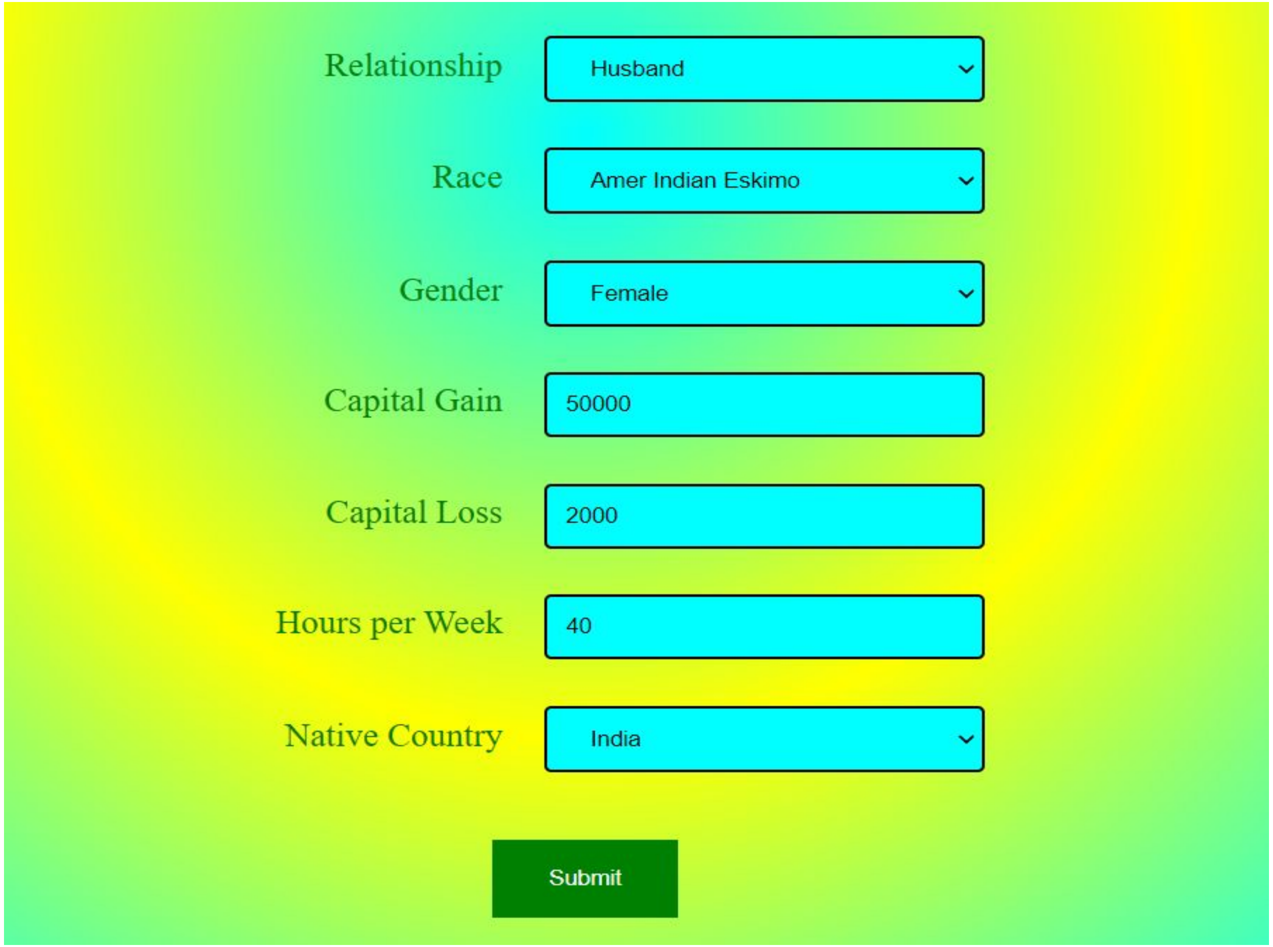
Code for results.html

```
<!doctype html>
<html>
    <body style="margin:auto;background-image: url('myimg1.jpg'); background
        -repeat: no-repeat; background-attachment: fixed; background-size: cover
        ;">
        <h1 style='width:100%;text-align:center; color:blue; margin-left:auto
            ;margin-top:200px;'> {{ prediction }}</h1>
    </body>
</html>
```

Execution Screenshots ---->>

Income Prediction Form

Age	21
Working Class	Local-gov
Education	Bachelors
Marital Status	not married
Occupation	Exec-managerial
Relationship	Unmarried


Relationship: Husband
Race: Amer Indian Eskimo
Gender: Female
Capital Gain: 50000
Capital Loss: 2000
Hours per Week: 40
Native Country: India
Submit

Have a caution as probably their income is less than 50K.

Income Prediction Form

Age: 43

Working Class: Federal-gov

Education: Bachelors

Marital Status: not married

Occupation: Exec-managerial

Relationship: Unmarried

Relationship	Unmarried
Race	Amer Indian Eskimo
Gender	Male
Capital Gain	90000
Capital Loss	0
Hours per Week	50
Native Country	India

Submit

Allow them to have a loan their income is more than 50K.


Thank you