Assignment-2

Neural-Network for "Census Income"

By: Khursheed Ali [163059009]

Code

Below is the brief description about code.

Code has "**class NN**" class stands of "Neural-Network", by creating the object of class neural network can be created, by defining following properties.

Properties of class NN

i. no_of_hidden_layer: Number of hidden layer in the network, default is value 0.
ii. neurons_per_hidden: number of nodes per hidden layer, default value is 1 node. It will be used with "no_of_hidden_layer".
iii. learning_rate: Value of alpha (learning parameter) in gradient descent, default value is set 1.
iv. lamda: Regularizer parameter, using L2 regularizer for controlling weights.
v. no_of_iteration: Number of iteration to be performed on neural network, default value is 10000 (ten thousand)
vi. enable_bias_per_hidden: It is a flag. If this value is true then on every hidden layer one bias node will be added. If setting says "neurons_per_hidden" layer is 10 and this flag is true then neural network will have 11 nodes per hidden layer.

Methods of class NN:

i. createNN(): When this method is called then neural network with the defined configuration and initializes the weight vector "**random**" weights.
ii. train(): This starts training of neural network. One can see the training log by enabling the property "enable_log".
iii. gradientdescent(): Perform the gd on the define loss function.
iv. activationFunction(): One can set activation function either "Sigmoid" or "ReLu"
v. forwardPropagation(datapoint): It will find the output of each node of nn, save these output in the "layer_output" property and returns the last layer output.
vi. backwardPropagation(layer_output): It takes the input as output of each layer and perform the back propagation and saves the result.
vii. updateWeightes(): It will update the weights according to the back propagation.
viii. predict(datapoints): It will predict the values of data points and return the predicted output but not thresholded output.
ix. getThresholdValue(input, threshold_value): It will do the thresholding of the input.
x. writeWeights(..): It will write calculated weights on to the file.
xi. loadWeights(filename); It will load weights from the passed file on the initial weight vector. This help in starting from olca calculated location.

Data manipulation:

i. Categorical column conversion: Code support two type of conversion:
    a. Range: Each category is given number from 1 to n. Used "get_dummy" and "argmax"

b. Column: For each category their will one column having value 0/1.

ii. Handling missing data **"?"**: Code has support two types for handling missing data.
   a. Leave as it is, i.e converting "?" to 0.
   b. Replace it with "**most frequent**" value.

iii. Data Normalization: By setting normalization true, data can be normalized. For normalization "divide by max(col)" is used.

iv. Drop column: By specifying the column names, columns can be dropped.

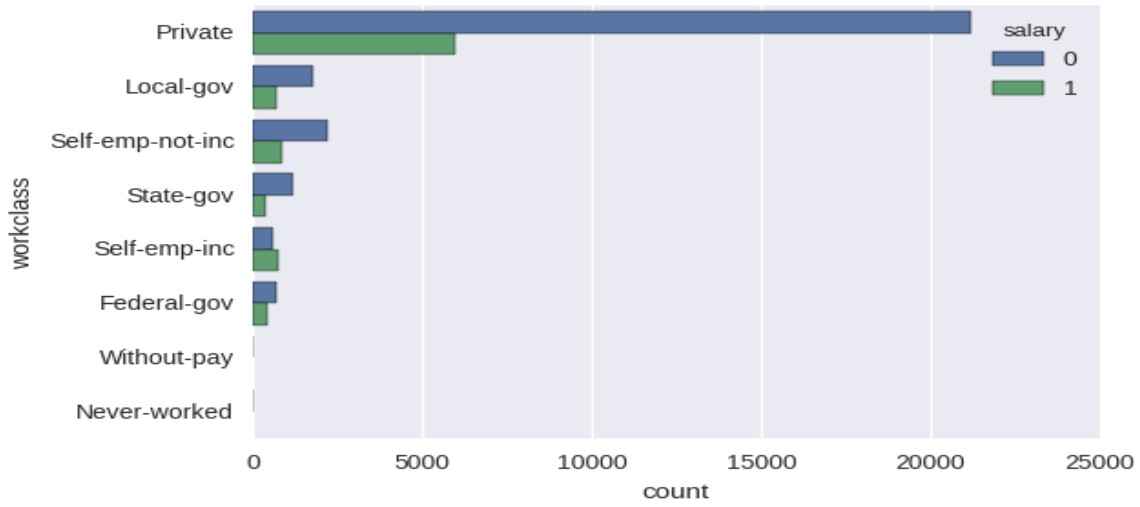
Working

i. Data is fetched using the padas library.
ii. From fetched data missing value "?" Are set as NaN.
iii. Then missing value is replace with "most frequent" entry in that column.
iv. Features are dropped if there are any.
v. Categorical data is then converted into column wise data using "get_dummies".
vi. Data is then normalized.
vii. After NN is created and data is given to network
viii. NN training is started
ix. After training is done, prediction is performed on test data

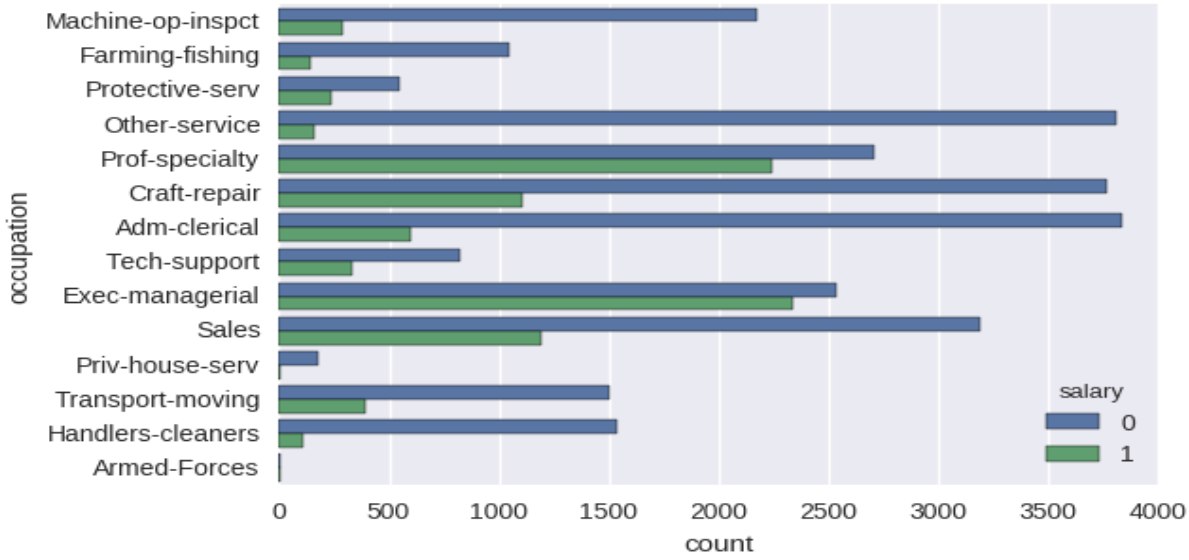
My Final Neural Network Settings

- Number of hidden layers: 2
- Neurons per hidden layer: 20
- Bias per hidden layer: True
- Learning rate: $1x10^{-4}$

- No of Iteration: 5000*10 = 50,000
- Categorical column: 2nd method I.e added column for each category
- Missing data "?" : Replaced the missing data with the most frequent value.
    - For "workclass" most frequent is "Private"



    - For "occupation" most frequent are "other-service", "craft-repair" and "Adm-clerical". For these three frequency count are almost same. So for replacing "?", by testing finally selected "craft-repair".



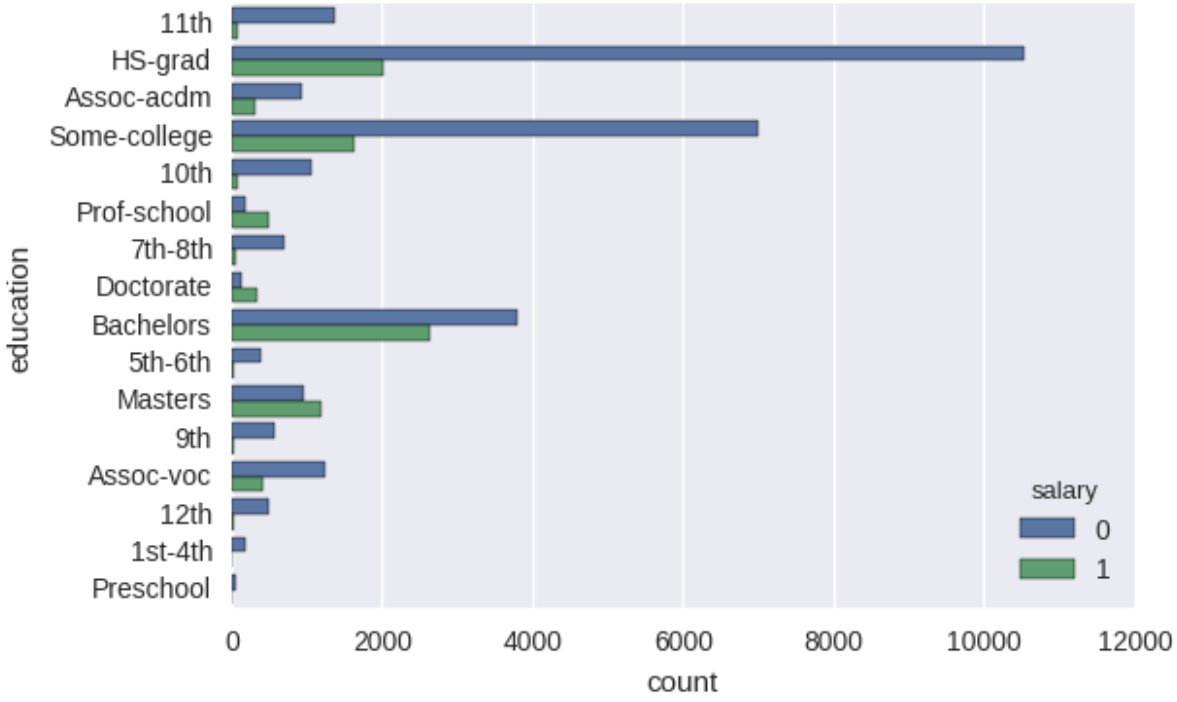
- For "native-country" most frequent is "United States". This can be seen from the "country" graph of Feature analysis section.

Features Analysis

1. Feature "education-num" and "education" are similar.

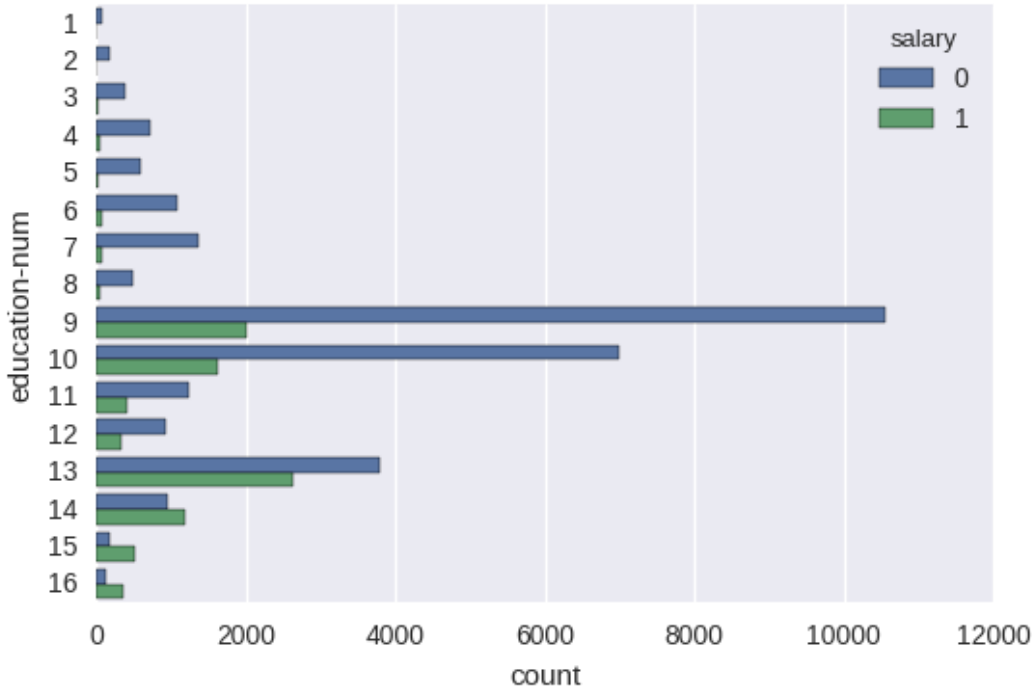
   A) Feature: **Education**

   Code: sb.countplot(y="education", hue='salary', data=trained_dataset)



B) Feature: **Education-Num**

Code: sb.countplot(y="education-num", hue='salary', data=trained_dataset)

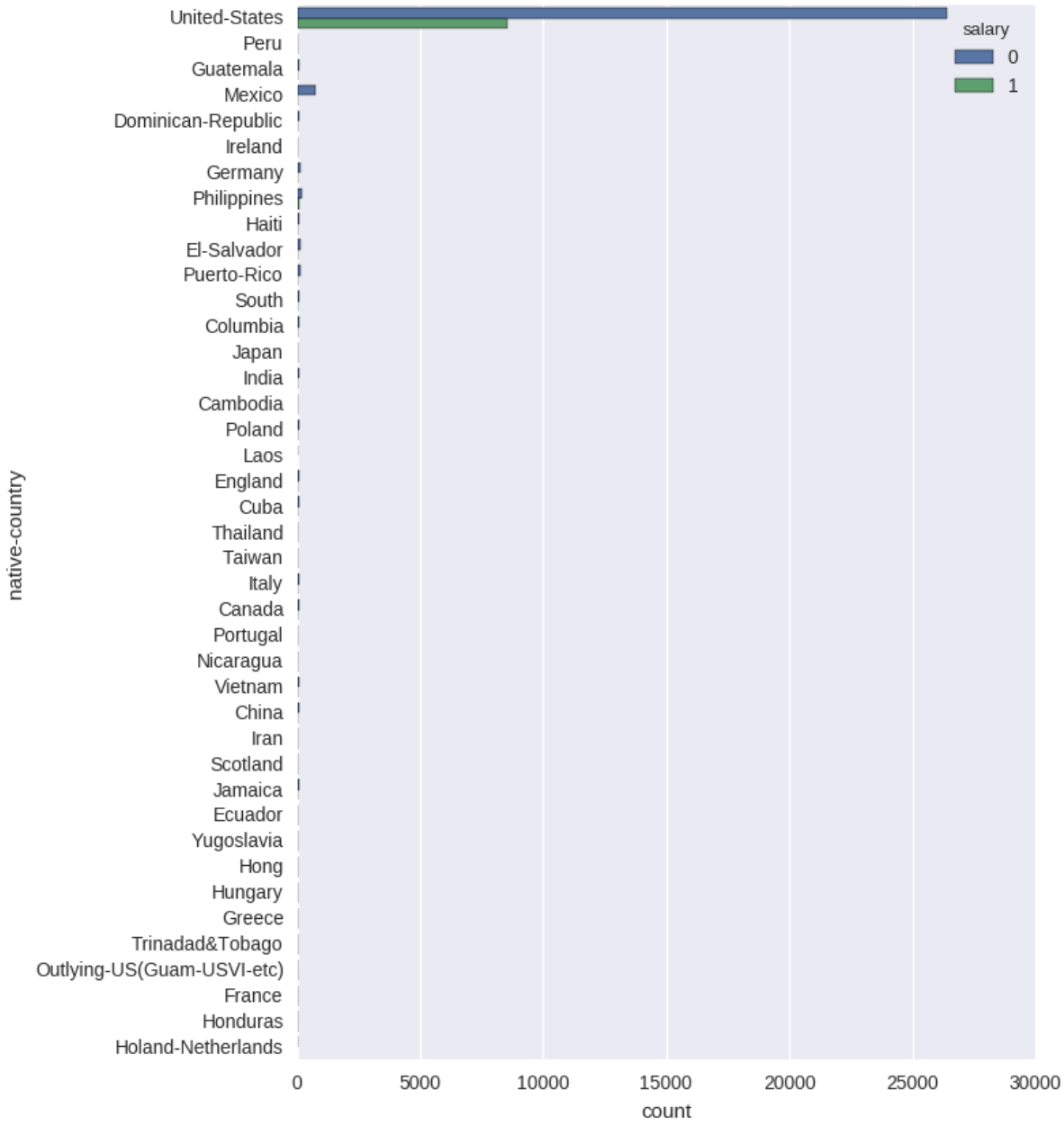


Both graph A and B are isomorphic. So we can remove one of the column from the data set before training the neural network. As in my implementation I'm adding column for each "category" rather than numbering the features from 1 to n, so this increasing the number of features and also the time for processing. To reduce the number column, I dropped "Education" column.

2. Feature "native-country"

Code: plt.figure(figsize=(7,10), dpi=300)

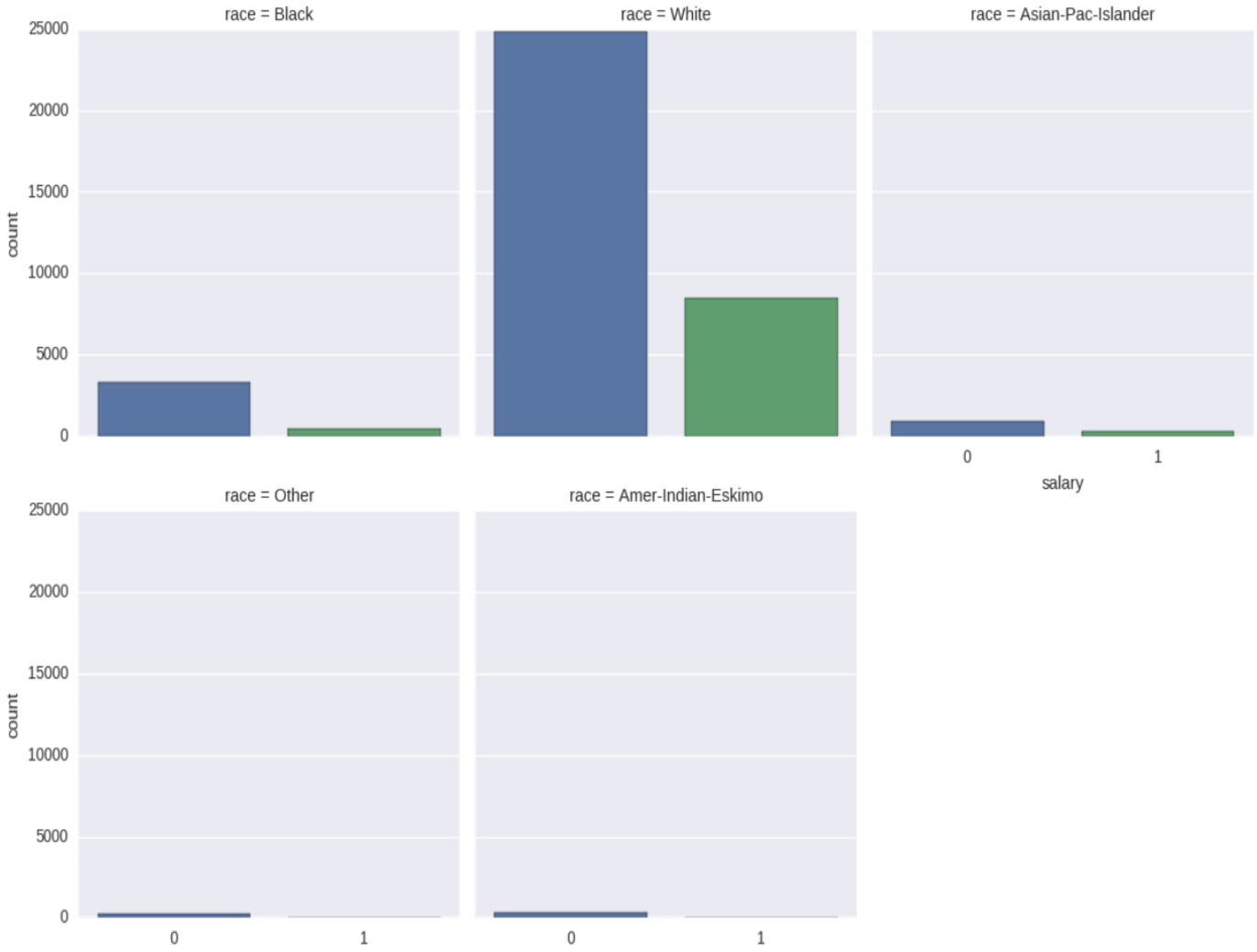
sb.countplot(y="native-country", hue='salary',data=trained_dataset)



From above graph one can see that the ratio to salary for "United-States" is pretty much more also for other remaining countries its very close. Means if native-country is something else other than "US" then it will be tough to classify based on "native-country". So I have removed this column from the feature dataset before training and testing as data is skewed towards the "united-states"

3. Feature "race"

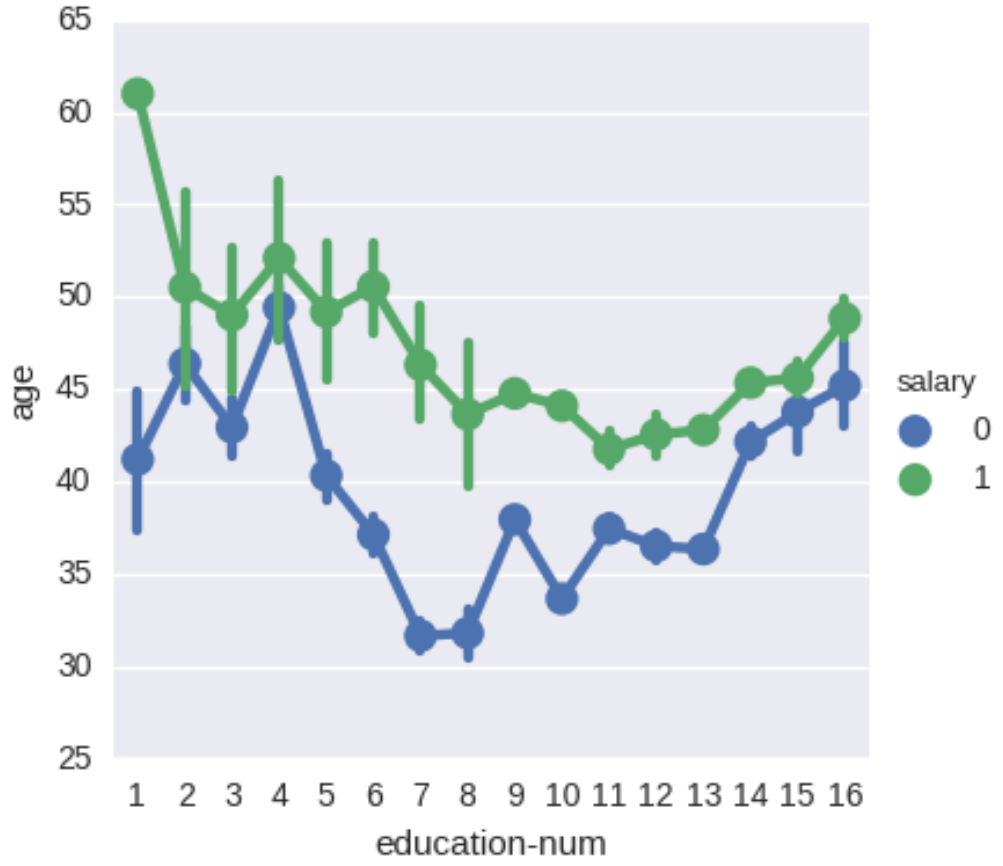
Code: sb.factorplot("salary", col="race",kind="count", col_wrap=3, data=trained_dataset)



As from the graph it can seen that for "Other", "Ammer-Indian", "Black" and "Asian" ratio of salary is very less and the record is very less. It is totally skewed towards "white", removing this feature or putting this feature was not affecting much. Therefore I removed this feature from the trained and test data.

4. Feature *correlation* between "age" , education-num" and "salary".

Code: sb.factorplot(x="education-num", y="age", hue="salary", data=trained_dataset)

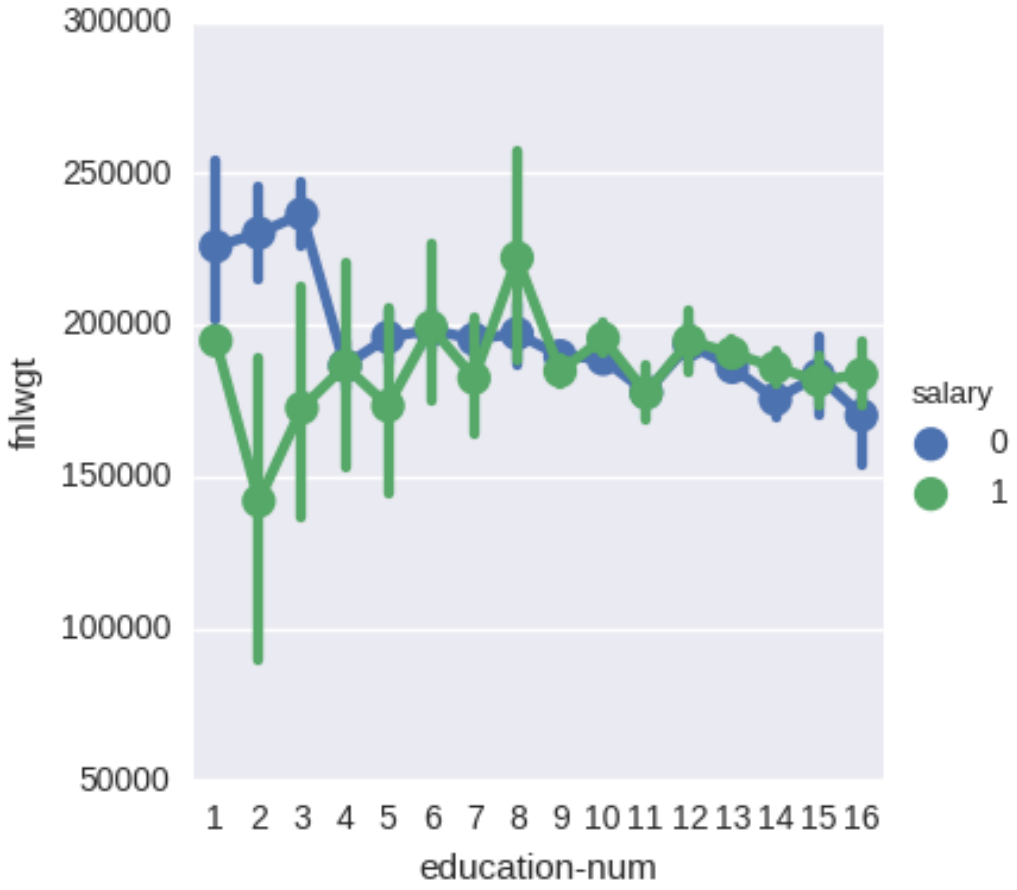


The height of this graph shows the mean and 95% confidence interval. Say for education level "8" and salary less than $50000, means *mean* age is around 33 and for salary greater than $50000 is around 44, in other words which tell that if the person has education of level "8" and his/her ages is nearby 30 to 35 then it's a high probability that he/she may be earning less than $50000 because of less experience in work, as his/her experience is increased with time he/she will get salary more than $50000. Similarly if for education level 16, mean age of salary less or more than $50000, is very close, therefore if we know the person's age it will be tough to say what will its salary.

For graph it is visible that education and age related to each other and can give good insight of person's salary if using these two features.

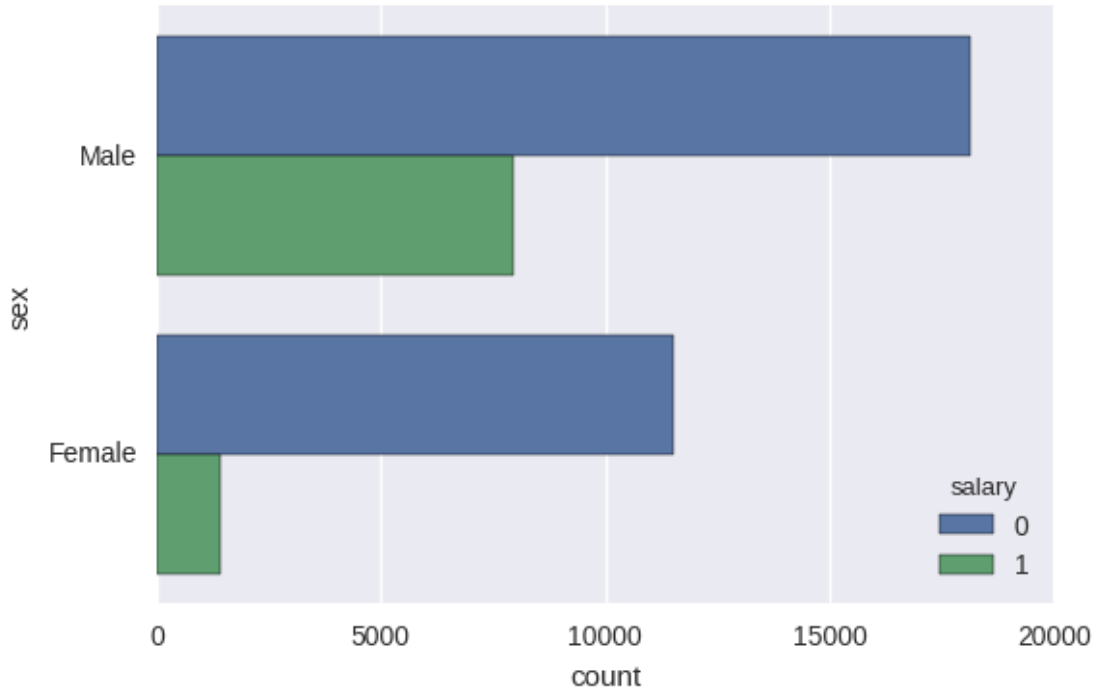
5. Feature *correlation* between "fnlwgt" , education-num" and "salary"

Code: sb.factorplot(x="education-num", y="fnlwgt", hue="salary", data=trained_dataset)



Here we can see how fnlwgt is correlated with education. Overall mean of fnlwgt is between 150000 to 200000 for approx all education, which show it's a essential feature

6. Feature "sex", it's not "skewed" as compare to other feature like "native-country". Therefore this may fill help in predicting better result.



Observation/Analysis

- Categorical data as range.
  - Number of features were 16 after adding bias column.
  - 4 Hidden layer + 60 neuron each layer: After training for 2 to 3 hrs I.e around 2 lakhs iteration it was giving error auc 0.67
  - 2 Hidden layer + 48 neuron each layer. After lots of training it was giving test auc 0.76 using sigmoid activation function
- Categorical data as column per category.
  - Number of features were 60 after adding bias column and dropping 2 columns "native-country" and "race".
  - 3 Hidden layer + 102 neuron each layer: It around 9 to 10 hrs to train for reaching to test auc 0.87.
  - Number of features were 44 after adding bias column and dropping 3 columns "native-country" , "race" and "education".

- 2 Hidden layer + 20 neuron each layer: It around only 1hrs to reach up to mean auc of 0.79 after training for much more time reaching up to 0.91

Observation: By training system with different number of features, hidden layer and neurons time of training the system varies lot. When number of neurons are less and hidden layer, then time to train is very less if we increase the neurons then it increase drastically as weights to trains are increased.

Result Comparison

Algorithm	Mean error [Train data]	Misclassified [Train data]	Accuracy [Train data]	AUC [Test data]
Neural Network	0.14486	5534	0.858004	0.79522
Logistic Regression	0.202422	7889	0.797578	0.60958
KNN	0.1636	6376	0.8364	0.62932
Gaussian NB	0.205578	8012	0.794422	0.63380

Mean error: np.mean(np.abs(y_pred - y))

As we can see that NN is giving best result out of other three classifier which are "Logistic Regression (LR)", "K Nearest Neighbor (KNN)" and "Gaussian Naive Bayes (GNB)" classification. Here after NN, KNN is giving the good performance after that GNB which is very close to the KNN and then LR.

So, by this we can conclude that Neural Network do best classification than other three.