TEAM 44 : Data Vaders

System Documentation Report

Income Analysis & Prediction

Members:

Karim Afifi

Ioana Tiriac

Frank Forelle

**Roles and responsibilities.**​

We had an initial data exploration phase, where we shared each other’s code over Google Collaboration platform. After debating what percentage of the project should be about data analysis using charts and how much we should focus on using a classifier and prediction model, we went through several approaches: Gradient Boosting classifier, Decision tree classifier and using an Artificial Neural Network for prediction purposes. Karim organized the meetings, did a lot of data exploration and initiated the proposals involving classifiers, Ioana focused more on data exploration, analysis & visualization and Frank also did data exploration & visualizations and built the ANN with high accuracy. After using the decision tree classifier, we had a ranking of most important features and could cross-reference our charts for each top 5 features and decided to include the best ones in the presentation. All team members were involved in creating the presentation and documentation and participated in all brainstorming meetings and reviews of each other's work.

**Team goals and a business objective.**

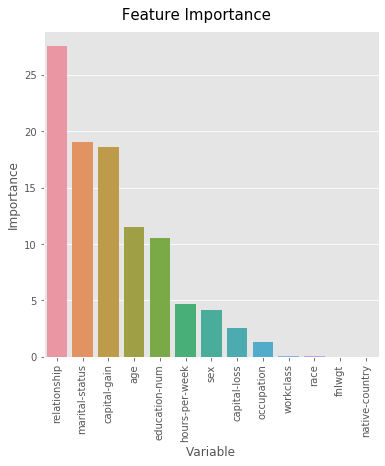
Our purpose is to help UVW College increase their enrollment by helping them identify proper individuals for their target group using a $50,000 threshold for the person’s income. We have therefore performed data analysis and deployed Machine Learning and AI techniques on the adult US Census data to determine which features have the greatest influence on an individual’s income, and what correlations exist between those features and income. We’ve also trained a neural network to predict the income of an individual with 84% accuracy by inputting the most important features of that person.

**Assumptions**​.

List of technical or business assumptions our team made :

* UVW College needs our professional help as data scientist to create marketing profiles based on people’s income
* UVW College wants to send appropriate enrollment proposals (pricing based) to individuals based on their income and is open to enrolling higher earning individuals as well as less than 50.0000$ earning ones
* UVW College is looking to develop an application to predict whether a person is in their target demographic, for that it also needs the most useful attributes of an individual
* UVW College is opened to proposals and suggestions regarding the creation of a prediction application

**Visualizations**​.

**Feature Importance**

After fitting a decision tree classifier on the training data, this figure was used to determine how important each feature is in constructing the decision tree. A total of 100 decision trees were constructed and the feature importance for each variable was calculated by taking the average of feature importances from the 100 decision trees. The reason for fitting 100 decision trees and taking the average is that if two features are highly correlated with each other and both have the same correlation with the target variable, the decision tree will pick one of them randomly which might decrease the feature importance value for some variables. Also, when deciding what feature to use for splitting the node, the algorithm chooses between a limited number of features and not all of them. For example, the algorithm might choose from sqrt(number of features). Those features are selected randomly and the best one will be picked for splitting. This can cause the feature importances to change between different models, and building multiple models and then taking the average will at least guarantee the persistence of the ordering of feature importances between different variables.

est = DecisionTreeClassifier()

param\_grid = {'criterion': ['gini', 'entropy'], 'splitter': ['best', 'random'], 'max\_leaf\_nodes': [2, 4, 5, 8, 10, 12, 14], 'max\_features': ['auto', 'sqrt', 'log2'], 'max\_depth': [5, 10, 15, 20]}

gs\_class = GridSearchCV(est, param\_grid, n\_jobs=-1, verbose=0, cv=5, scoring='balanced\_accuracy')

res\_cv=gs\_class.fit(X\_train\_hot\_enc, y\_train)

feature\_importances=[]

for i in range(100):

est = DecisionTreeClassifier(\*\*res\_cv.best\_params\_)

est.fit(X\_train, y\_train)

feature\_importances.append(est.feature\_importances\_\*100)

feature\_imp\_final=[[], [], [], [], [], [], [], [], [], [], [], [], []]

for fi in feature\_importances:

for i in range(len(fi)):

feature\_imp\_final[i].append(fi[i])

feature\_imp\_final\_temp=[]

for fi in feature\_imp\_final:

feature\_imp\_final\_temp.append(np.mean(fi))

feature\_imp\_final=feature\_imp\_final\_temp

del feature\_imp\_final\_temp

fi\_df = pd.DataFrame({'Variable': X\_train.columns, 'Importance': feature\_imp\_final})

fi\_df.sort\_values('Importance', axis=0, ascending=False, inplace=True)

plt.figure(figsize=(6,6))

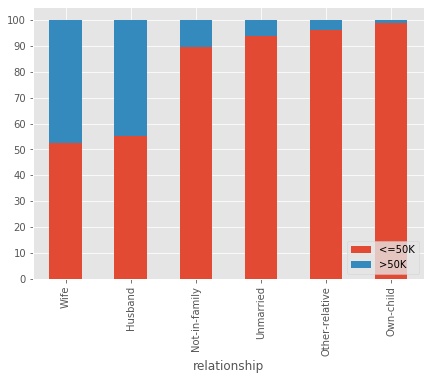
plt.title('GB Feature Importance', fontsize=15, pad=12)

sns.barplot(fi\_df['Variable'], fi\_df['Importance'])

plt.xticks(rotation=90)

The top 5 features were then selected for deeper analysis and to understand the relationship between those features and income. The top 5 features were also considered for the executive presentation. Figures were plotted for the relationship between all features and income but the top 5 features will only be provided in this document for space efficiency. Stacked bar charts and mosaic plots were used to visualize and assess the relationship between a categorical feature and categorical target variable (income). Box plots and multi-histogram plots were used to visualize and assess the relationship between a numerical feature and a categorical target variable (income). Selecting the visualization type was based on being the right approach to visualize the participating features’ data types, the message we wanted to deliver and the clarity of the visualization.

**Relationship Status**



The above visualization shows the income tendency of people in a certain relationship. The most prominent data trend here is that married individuals make more than 50K compared to other individuals in a different kind of relationship. UVW college can therefore adjust their marketing efforts to take the relationship status attribute of a person into consideration when issuing an enrollment proposal.

rppct=pd.crosstab(train\_data['relationship'], train\_data['Income']).apply(lambda r: r/r.sum(), axis=1)\*100

rppct.sort\_values(by='>50K', ascending=False, inplace=True)

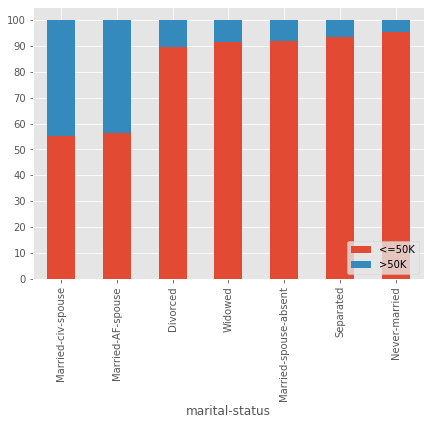
rppct

rppct.plot.bar(stacked=True, figsize=(7, 5))

plt.legend(loc='lower right')

plt.yticks(list(range(0, 101, 10)))

**Marital Status**



As the second most important feature, the marital status demographic attribute confirms the same data trend as the relationship status, setting married people and approximately 10% of people with another commitment status in the list of high incomers (>50.000$). This enforces the idea that UVW college should take the marital status demographic attribute of an individual into consideration when issuing an enrollment proposal.

msct=pd.crosstab(train\_data['marital-status'], train\_data['Income']).apply(lambda r: r/r.sum(), axis=1)\*100

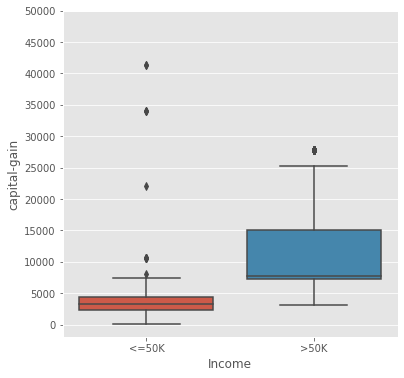
msct.sort\_values(by='>50K', ascending=False, inplace=True)

msct

msct.plot.bar(stacked=True, figsize=(7, 5))

plt.legend(loc='lower right')

plt.yticks(list(range(0, 101, 10)))

**Capital Gain**

Individuals who have a higher income than 50.000$ tend to have higher capital gains than people who do not. The capital gain attribute of an individual can be used by the UVW college to target individuals with suitable enrollment proposals, as 75% of higher earning adults (over 50.000$) have a higher than 7500 capital gain and nearly all of individuals who make less than 50k have a capital gain of less than 7500.

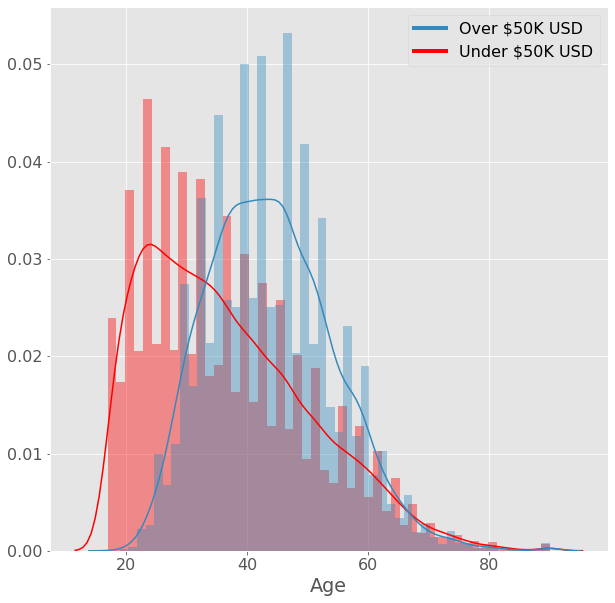
temp\_df=train\_data[['capital-gain', 'Income']]

temp\_df=temp\_df[(temp\_df['capital-gain']!=0) & (temp\_df['capital-gain']!=99999)]

plt.figure(figsize=(6,6))

sns.boxplot(temp\_df['Income'], temp\_df['capital-gain'])

plt.yticks(list(range(0, 50001, 5000)))

**Age** 

Age is the 4th most important feature we found to be influential on the income. This graphic shows that the average age of an individual earning more than 50.000$ is 44 years old, while the average age of those earning less than 50.000$ is 37. This can be a strong indicator for UVW college to target individuals younger than 37 years with appropriate enrollment programs appropriate for individuals earning less than 50K and individuals older than 44 years old with appropriate enrollment programs for individuals earning more than 50K.

custom\_red = (0.886,0.29,0.2,1.0)

custom\_blue = (0.204,0.541,0.741,1.0)

plt.style.use('ggplot')

plt.rcParams.update({

"figure.facecolor": (0.0, 0.0, 0.0, 0.0),

"font.size": 16

})

fig, ax = plt.subplots(figsize=(10,10))

print(col,"vs. Income (Red = under 50K, Blue = over 50K)")

fig\_dist1 = sns.distplot(df.loc[df['income'] == '<=50K'][col], ax=ax, color='red')

fig\_dist2= sns.distplot(df.loc[df['income'] == '>50K'][col], ax=ax, color=custom\_blue)

custom\_lines=[pltlines.Line2D([0], [0], color=custom\_blue, lw=4), pltlines.Line2D([0], [0], color='r', lw=4)]

plt.xlabel("Age")

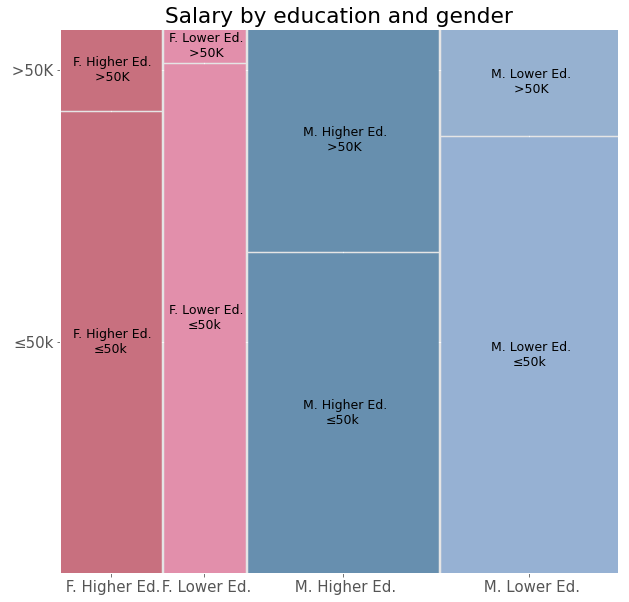
plt.legend(custom\_lines, ['Over $50K USD', 'Under $50K USD'], fancybox=True)

plt.show()

# to calculate mean values

df[df['Income']=='>50K']['age'].mean()

df[df['Income']=='<=50K']['age'].mean()

**Education Level**

We chose to display the 5th most important feature, education, in correlation with gender and income using a mosaic plot. From this graphic we can clearly see the majority of the individuals in the dataset are male and can draw the conclusion that regardless of gender, individuals with higher education ('Some-college', 'Bachelors', 'Assoc-acdm', 'Assoc-voc', 'Masters', 'Doctorate') have a higher percentage of individuals earning over 50.000$. This graphic also confirms our initial feature importance ranking with the gender feature not having as significant an impact on income. UVW College can use this trend to target individuals with higher education regardless of their gender with programs suited for high earners (over 50.000$) and individuals with lower education (at most high school) regardless of their gender with education offers suited for individuals making less than 50.000$.

df = pd.DataFrame.copy(data[['salary-class', 'gender', 'education']])

fig, ax = plt.subplots(figsize=(10,10))

higherEducList = ['Some-college', 'Bachelors', 'Assoc-acdm', 'Assoc-voc', 'Masters', 'Doctorate']

df['education'] = df['education'].map(lambda r: 'Higher Ed.' if str(r).strip() in higherEducList else 'Lower Ed.')

df['salary-class'] = df['salary-class'].map(lambda r: r if str(r).strip() == ">50K" else "≤50k")

df['combined'] = df['gender'] + df['education']

df['combined'] = df['combined'].map(lambda r: str(r).replace('Male', 'M. ').replace('Female', 'F. '))

f = df.sort\_values(by=['combined'])

cols = {(' F. Higher Ed.', ' >50K'):'#C8707E',(' F. Higher Ed.', '≤50k'):'#C8707F',

(' F. Lower Ed.',' >50K' ):'#E28FAB', (' F. Lower Ed.', '≤50k'):'#E28FAB',

(' M. Higher Ed.', ' >50K'):'#678FAE',(' M. Higher Ed.', '≤50k'):'#678FAF',

(' M. Lower Ed.',' >50K' ):'#96B1D0', (' M. Lower Ed.', '≤50k'):'#96B1D3'}

mosaic(df, ['combined','salary-class'], properties = lambda key: {'color': cols[key]}, ax=ax)

plt.rcParams['font.size'] = 15.0

plt.title("Salary by education and gender")

plt.show()

**Decision Tree**

Visualizing the decision tree can provide insights on what combination of features and their values lead to determine the income of an individual. While the team tried to use gradient boosting at the beginning, it was very hard to interpret hundreds of trees even if they are shallow ones. Therefore, we decided to pursue a decision tree for better interpretation. While a limit on the depth and number of leaf nodes of the tree was imposed to constrain the number of levels in a tree for easier visualization, the decision tree was still too large to be visualized in a few slides. Therefore, it is only provided in this documentation. Different parameters were tried to achieve the optimal accuracy while limiting its growth. After identifying the top important features in building the decision tree classifier, the top 5 features were then selected for further analysis. The top 5 features were also selected to build the final decision tree that shows the combination of variables and features that lead to selecting a certain class (>50k, <=50k). Selecting the top 5 features only for building the final decision tree and imposing more constraints on the depth and number of nodes of the tree allows the tree to be interpretable while sacrificing some prediction power. The prediction performance of the tree becomes less (around 80%) from 85% after imposing the constraints and using only the top 5 features.

X\_train=X\_train[list(fi\_df[:5]['Variable'].values)]

X\_test=X\_test[list(fi\_df[:5]['Variable'].values)]

est = DecisionTreeClassifier(\*\*res\_cv.best\_params\_)

est.fit(X\_train\_hot\_enc, y\_train)

dot\_data = StringIO()

export\_graphviz(est, out\_file=dot\_data,

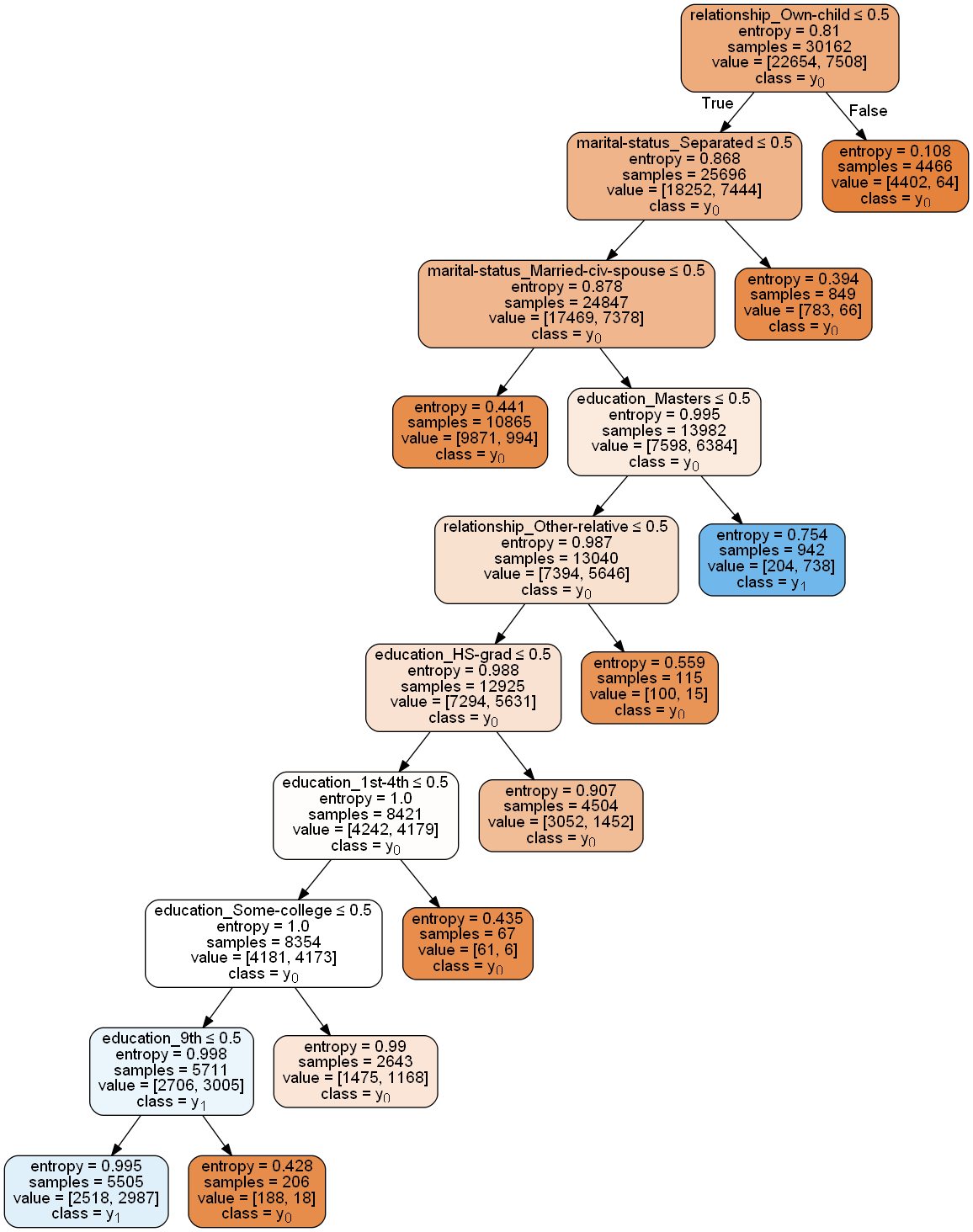
filled=True, rounded=True,

class\_names=True, feature\_names=list(X\_train\_hot\_enc.columns.values),

special\_characters=True)

graph = pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())

Image(graph.create\_png())



Categorical variables in our data required to be transferred to numerical variables using one hot encoding. The reason behind using one hot encoding is that the tree can then be interpreted easier. A hot encoded variable will result in variables equal to the number of categories within that variable. For example, if marital-status feature has 3 categories (Married-civ-spouse, Married-civ-husband, Separated), the original marital-status variable will be deleted and we will have 3 new variables: marital-status\_Married-civ-spouse, marital-status\_Married-civ-husband and marital-status\_Separated. Each new variable will be a binary variable consisting of 0s and 1s only. 1 corresponds to true while 0 corresponds to false. Now, we can interpret categorical variables easily in the decision tree. If a node splits on education\_Masters for example and it splits on <= 0.5, that means if the value of education\_Masters is 0, if the value is splits on is >0.5, that means if the value of education\_Masters is 1. The decision tree keeps doing binary checks on the variables to determine the final class of a certain sample. An advantage of decision trees is that they are highly interpretable by senior levels and non-technical individuals who want insights and knowledge besides prediction accuracy.

dot\_data = StringIO()

export\_graphviz(est, out\_file=dot\_data,

filled=True, rounded=True,

class\_names=True, feature\_names=list(X\_train\_hot\_enc.columns.values),

special\_characters=True)

graph = pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())

Image(graph.create\_png())

**Income Prediction by training an Artificial Neural Network**

ANNs were selected to predict income with the highest possible accuracy. ANNs are very powerful models that can capture complex relationships and therefore might result in a very good accuracy compared to other models, while at the same time requiring less insight into which variables are more or less important to accurately predict the class of an input.

Before fitting the neural network model, the dataset was first cleaned and normalized as normalizing the data highly increases the prediction accuracy of the neural network model and allows for faster convergence. One-hot encoding was used as with the decision tree to convert the categorical variables to numerical ones. Different combinations of number of layers, number of nodes, activation functions and other model parameters were tried to achieve better prediction accuracy.

# normalize age, fnlwgt, capital-gain, capital-loss, hrs-per-week

for col in ['age','fnlwgt','capital-gain','capital-loss','hrs-per-week']:

df[col] = df[col].astype(np.float32)/np.max(df[col].astype(np.float32))

Achieving one-hot encodings of the features :

df = df.join(pd.get\_dummies(df['workclass'], prefix='workclass'))

df = df.join(pd.get\_dummies(df['education'], prefix='education'))

df = df.join(pd.get\_dummies(df['marital-status'], prefix='marital-status'))

df = df.join(pd.get\_dummies(df['occupation'], prefix='occupation'))

df = df.join(pd.get\_dummies(df['relationship'], prefix='relationship'))

df = df.join(pd.get\_dummies(df['race'], prefix='race'))

df = df.join(pd.get\_dummies(df['sex'], prefix='sex'))

df = df.join(pd.get\_dummies(df['native-country'], prefix='native-country'))

df = df.join(pd.get\_dummies(df['income'], prefix='income'))

Next the data was split for training/testing purposes :

df\_train = df.drop('income\_>50K',axis=1)[:30162]

df\_train\_labels = df['income\_>50K'][:30162]

df\_test = df.drop('income\_>50K',axis=1)[30162:]

df\_test\_labels = df['income\_>50K'][30162:]

For the training of the ANN we’ve used a Sequential Keras model with a network structure consisting of a single ‘dense’, fully connected layer with 128 outputs for which we used the rectified linear unit activation function (ReLu). Compiling the model uses TensorFlow as the backend technology, that decides how the neural network is built on top of the hardware and the “Adam” **optimizer** as the efficient stochastic gradient descent algorithm. Next, we trained the ANN against the dataset across 20 epochs (iterations) to improve the prediction capability of the algorithm while preventing ‘overfitting’. Manually adjusting the epochs to larger magnitudes provided no meaningful gains in accuracy.

model = keras.Sequential([keras.layers.Dense(128, activation='relu'),

keras.layers.Dense(1)])

model.compile(optimizer='adam', loss=tf.keras.losses.BinaryCrossentropy(from\_logits=True),metrics=['accuracy'])

history = model.fit(df\_train, df\_train\_labels, epochs=20, verbose=0, callbacks=[tfdocs.modeling.EpochDots()])

The testing phase of the neural network showed that we have achieved a fit accuracy of aprox. 85% and test accuracy of approx. 84%, meaning that the ANN can predict the income of an individual with a probability of about 84% given the features of US census data.

test\_loss, test\_acc = model.evaluate(df\_test, df\_test\_labels, verbose=2)

print('\nTest accuracy:', test\_acc)

hist = pd.DataFrame(history.history)

hist['epoch'] = history.epoch

fit\_plot = sns.lineplot(hist['epoch'],hist['accuracy'])

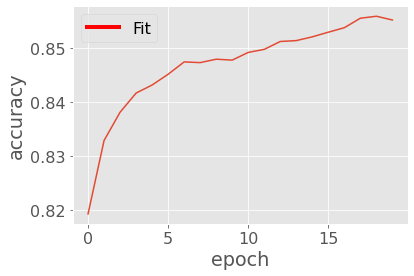
plt.legend([pltlines.Line2D([0], [0], color='r', lw=4)],['Fit'])

plt.figsize(6,6)

plt.show()

===========================================================================

Test accuracy: 0.8330677151679993

**Questions**​

* *How do we determine the features that influence income?*

The team decided to visualize the relationship between each feature and the target variable (income) to determine what features are strongly correlated with income and in what way. Also, tree-based models were used as a check on the feature importance of each variable. Feature importance score indicates how useful a variable was in the construction of the tree.

* *How do we determine the combination of features and values that can be used to decide the income of an individual?*

A classification based classifier was used to fit the training data since it provides high interpretability but less prediction accuracy. If we visualize a decision tree after building it, we can determine the combination of features and values used to reach a decision (class). The growth of the tree and the number of nodes were constrained to make it possible for visualization as a tree can grow to hundreds of levels if left constrained.

* *How do we predict income with the highest possible accuracy?*

A Neural network model was used as it is one of the most powerful machine learning models and can capture very complex relationships and it will probably provide the best accuracy.

**Potential for further investigation**

Segmentation might also be helpful to understand what features can help determine income. A clustering model like k-means can be used to cluster all the individuals into two clusters corresponding to the two different classes (>50k, <=50k). An accurate segmentation model would result in two clusters which are pure. A pure cluster contains individuals only from one class. The higher the purity of the two clusters and the better the model can differentiate between the two classes, the better the segmentation model. Different combinations of features can be tested until reaching two clusters with the highest purity. The combination of features that lead to the highest purity can be considered as the most important features as they can be used to separate the two classes accurately. Investigating the range of values for each feature in the two separate clusters can be very insightful. For example, if the range of the age variable in cluster one that contains a majority from class one is 18-30 and the range of the same variable in cluster two that contains a majority from class two is 28-45, we can conclude that individuals from class two tend to be older than class one. Segmentation is another approach to solve the same problem and it would be interesting to examine such an approach in the future.

Additionally, there is still work to be done in optimizing the ANN. Additional layers, activation functions and optimizers may result in higher accuracy prediction capabilities. Also, the model slightly overfits currently and tweaking the parameters may alleviate this issue.