Mark McCarthy: S395E958
Nithin Reddy Nagapur: J694D927
-
-
-
-
CS697-AB Final Project

Salary Classification

Problem Statement

The task our group was assigned was to classify whether a person's salary is above or below $50,000 based on our given data set. To accomplish this we were allowed any means we had covered in our course along with any relevant deeper studies we had performed outside of class.

Data

Our data set consisted of fifteen features, namely ['age', 'workclass', 'fnlwgt', 'education', 'education-number', 'marital-status', 'occupation', 'relationship', 'race', 'sex', 'capital-gain', 'capital-loss', 'hours-per-week', 'native-country', 'salary'], and 32561 rows. The nonnumeric features cover a substantial range of representative data. The 'workclass' feature covers 9 categories of employment sectors such as 'Private', 'Federal-gov', 'Self-emp-inc', etc.. For the 'fnlwgt' it was determined to not have a strong enough correlation to keep. This will be mentioned in more detail later in this section. Education ranged from preschool to Doctorate on the US scholastic system with the addition of trade school under the label 'Prof-school'. Occupation covers various industry categories instead of individual job titles for generalization. The other features are relatively self-explanatory.

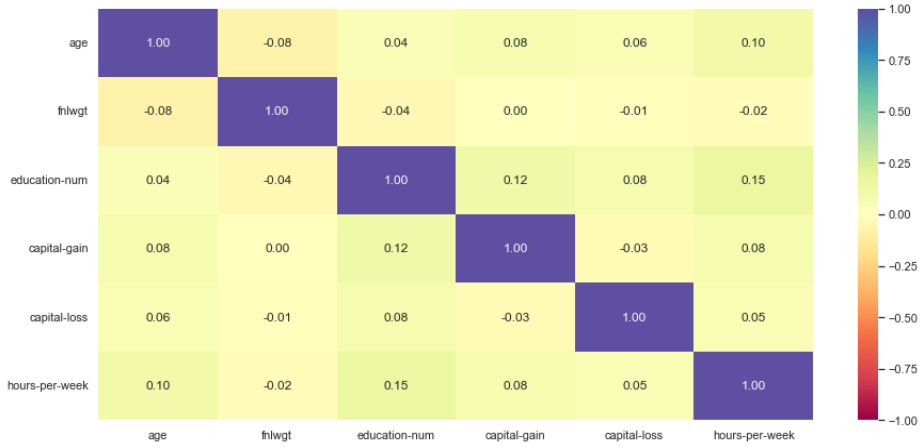
The cleaning necessary for the data set involved removing various "?" entries for rows in 'native-country', 'workclass', and 'occupation'. These entries were dropped. The 'flnwgt' feature proved negatively related in a majority of a heat map derived from the df.corr() preset which computes the pairwise correlation of features resulting in a decision to drop the feature entirely (1). It was determined that 'education-number' and 'education' were representative of each other. This was shown

by stacked bar graphs created in the EDA phase of the project (2a)(2b). From this the numeric representation was kept for ease of use and the string representation was dropped. Outliers were handled by taking no action as they proved highly relevant to the data through box-plots of all numeric features. Since the data in this set are so close in scale to each other, no scaling or normalization was necessary. This concluded the data manipulation and preprocessing.

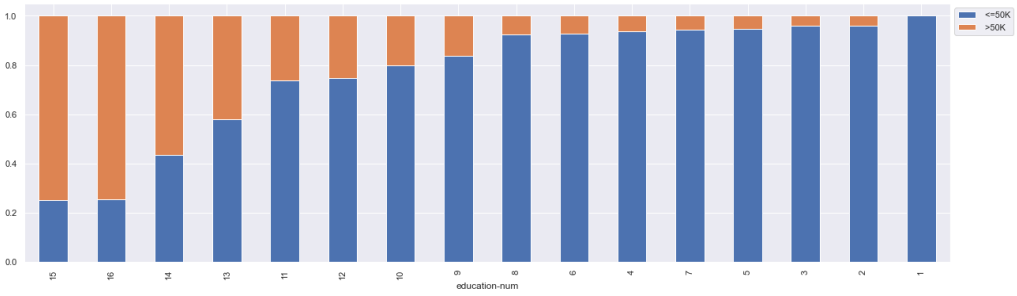
Methods

We began our EDA with uni-variate analysis. We developed a function to plot a box-plot and histogram along the same scale of feature and count for each of the numeric features in the data set which yielded ‘capital-gain’ and ‘capital-loss’ to be right skewed. Also the mean and median for ‘age’, ‘education-number’, and ‘hours-per-week’ to be relatively similar. From here we moved to bi-variate analysis for the non-numeric features, again looking at the count vs feature bar graphs. A function to output this graph was developed for the bi-variate analysis as well. From these bar graphs it was determined that for ‘workclass’, ‘Private’ made up the majority of the entries, ‘education’ was dominated by ‘HS-grad’, ‘occupation’ was varied with ‘Craft-repair’, ‘Exec-Managerial’, and ‘Prof-specialty’ topping the graph(3), ‘marital-status’ consisted mainly of ‘Married-civ-spouse’, ‘Never-married’, and ‘Divorced’, ‘relationship’ was largely ‘Husband’, ‘race’ was predominantly ‘White’, ‘sex’ was predominantly ‘male’, and ‘native-country’ was so dominated by ‘US’ that we decided to represent this feature with ‘US’ or ‘Other’. Our target feature of ‘salary’ was 24.89% above 50K. From here we plotted the pairplot() method from seaborne python package which plots all numeric features in the data frame vs each other in a grid. No notable information was drawn from this. A sampling of graphs from our EDA are present below. This sampling is representative of the treatment of all features but most were omitted from the report to avoid unnecessary length. This concluded our EDA process.

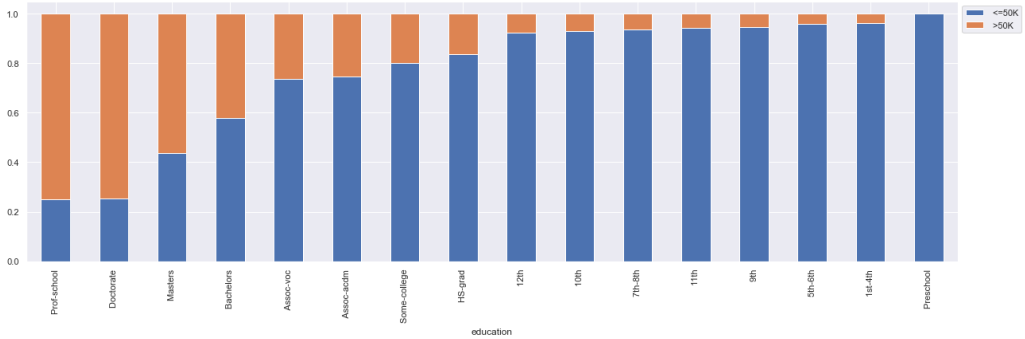
(1)


age
fnlwgt
education-num
capital-gain
capital-loss
hours-per-week
1.00 -0.08 0.04 0.08 0.06 0.10
-0.08 1.00 -0.04 0.00 -0.01 -0.02
0.04 -0.04 1.00 0.12 0.08 0.15
0.08 0.00 0.12 1.00 -0.03 0.08
0.06 -0.01 0.08 -0.03 1.00 0.05
0.10 -0.02 0.15 0.08 0.05 1.00
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

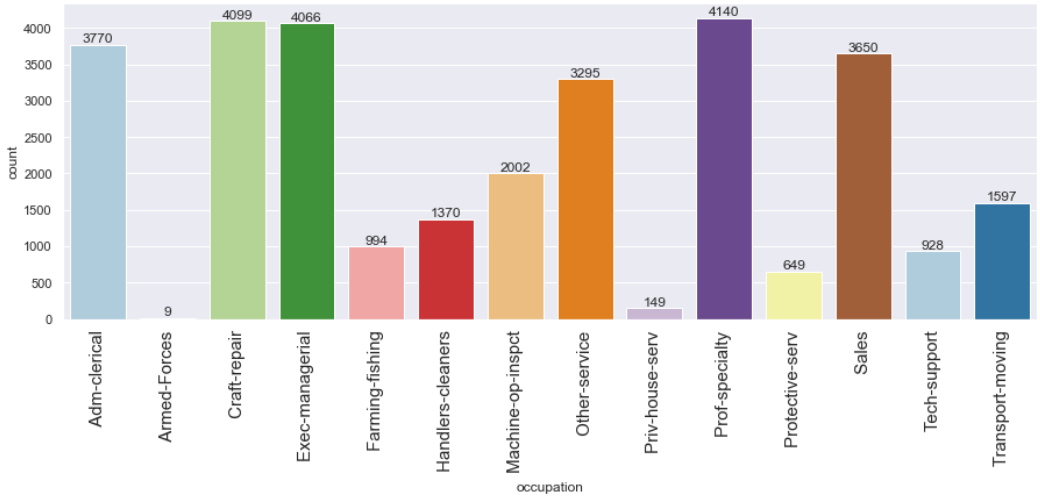
(2a)


<=50K
>50K
1.0
0.8
0.6
0.4
0.2
0.0
15 16 14 13 11 12 10 9 8 6 4 7 5 3 2 1
education-num

(2b)


<=50K
>50K
1.0
0.8
0.6
0.4
0.2
0.0
Prof-school
Doctorate
Masters
Bachelors
Assoc-voc
Assoc-acdm
Some-college
HS-grad
12th
10th
7th-8th
11th
9th
5th-6th
1st-4th
Preschool
education

(3)



We now moved into the model building. Our approach was to individually test the performance of multiple types of classifiers. We tested decision trees, random forests, bagging classifier, gradient boosting classifier, and SVM. Two functions were developed to concisely handle each of the models and generate a confusion matrix for each. The model assessing function as well as the confusion matrix generator took as inputs the classifier, predictors, and target. From here we modified the data frame in preparation for training by converting the 'salary' feature to a binary representation of 1 if greater than or equal to 50k and 0 if not via lambda function. The data frame was then split into 'X' and 'Y' (Training data and targets respectively) where X was the data frame less the 'salary' feature and Y was the 'salary' feature. Then we utilized the train test split built in method with 70-30 split, random_state =1, stratify = y. The preparation of the pre-training data was verified to have the proper shape and contents and we began the model training. For each of the classifiers a model was assigned and fit then passed to our developed model functions and outputs reviewed. The metrics chosen to asses the models performance were accuracy, recall, precision, and F1 score output as a data frame and the Sklearn confusion matrix method. After reviewing the metrics GridSearchCV was used to tune the hyper parameters of each model. For decision tree classifier we used a parameter grid of :

"max_depth": np.arange(10, 30, 5),

"min_samples_leaf": [3, 5, 7],

"max_leaf_nodes": [2, 3, 5],

"min_impurity_decrease": [0.0001, 0.001],

for random forest classifier:

"max_depth": list(np.arange(5, 15, 5)),

"max_features": ["sqrt", "log2"],

"min_samples_split": [3, 5, 7],

"n_estimators": np.arange(10, 40, 10),

for bagging classifier:

"max_samples": [0.7, 0.8, 0.9],

"max_features": [0.7, 0.8, 0.9],

"n_estimators": np.arange(90, 120, 10),

for gradient boosting:

"n_estimators": [200, 250, 300],

"subsample": [0.8, 0.9, 1],

"max_features": [0.7, 0.8, 0.9, 1],

"learning_rate": np.arange(0.1, 0.4, 0.1),

and for SVM:

"C":[1, 10, 100, 200, 500],

"gamma": [1, 0.1, 0.01, 0.001, 0.0001],

"kernel": ['rbf'],

after tuning each model was reassessed and compared to all others via a grid of the output metrics. After all models had been tuned and assessed, we decided to review which dummy features were most influential by using the built in feature_importances_ method. The top 15 features of the two best

performing models were then selected to run a second round of training predicting. The same method as the original data set was used with the new sets of features which were labeled as alpha and beta. This concluded our experimentation with this data set.

Results

The best performing model we were able to generate for the original data set was the random forest classifier after tuning with an accuracy of ~86% and a recall of ~96%. Close to this we achieved similar results in multiple other models as shown in (4).

(4)

	Decision Tree	Tuned Descision Tree	Random Forest	Tuned Random Forest	Bagging Classifier	Tuned Bagging Classifier	Gradient Boost Classifier	Tuned Gradient Boost Classifier	SVM Classifier	Tuned SVM Classifier
Accuracy	0.810808	0.717759	0.847055	0.858658	0.834567	0.859653	0.869599	0.872583	0.861200	0.872030
Recall	0.863911	0.656760	0.910990	0.960424	0.890246	0.931587	0.949978	0.945564	0.969987	0.949537
Precision	0.881814	0.952828	0.888251	0.866012	0.889591	0.887208	0.884884	0.891401	0.862394	0.887880
F1	0.872771	0.777565	0.899477	0.910778	0.889918	0.908856	0.916276	0.917684	0.913031	0.917674

the best performance seen for the alpha data set was determined to be the tuned SVM classifier with accuracy of ~85% and recall of ~97% which indicates that trimming the original data set of the less important features was in fact beneficial albeit slightly. The performances of all models for the alpha set is shown in (5)

(5)

	Tuned Random Forest	Tuned Random Forest Alpha	Tuned Random Forest Beta	Tuned Gradient Boost Classifier	Tuned Gradient Boost Classifier Alpha	Tuned Gradient Boost Classifier Beta	SVM Classifier	Tuned SVM Classifier Alpha	Tuned SVM Classifier Beta
Accuracy	0.857576	0.863023	0.866765	0.875716	0.868185	0.870175	0.858334	0.865533	0.854829
Recall	0.958189	0.953774	0.956991	0.946018	0.946144	0.944630	0.968784	0.960837	0.969099
Precision	0.866347	0.875080	0.876864	0.894567	0.886074	0.889384	0.860231	0.872923	0.856482
F1	0.909957	0.912734	0.915177	0.919573	0.915124	0.916175	0.911286	0.914773	0.909317

for the beta set the result was the same as the alpha set with the tuned SVM classifier performing the best. The results of this set showed an increase across the board yielding the best results of the project with accuracy ~86%, recall ~97%, precision ~86%, and F1 of ~0.912. The comparison of the models performing on the beta set is shown in (6). It was also noteworthy that the most important feature for having a salary greater than 50K was determined to be ‘marital status’.

(6)

	Tuned Random Forest	Tuned Random Forest Alpha	Tuned Random Forest Beta	Tuned Gradient Boost Classifier	Tuned Gradient Boost Classifier Alpha	Tuned Gradient Boost Classifier Beta	SVM Classifier	Tuned SVM Classifier Alpha	Tuned SVM Classifier Beta
Accuracy	0.858658	0.859985	0.863410	0.872583	0.869046	0.872804	0.861200	0.867831	0.858990
Recall	0.960424	0.953656	0.956746	0.945564	0.948507	0.947771	0.969987	0.962778	0.971311
Precision	0.866012	0.871940	0.873472	0.891401	0.885334	0.890025	0.862394	0.874048	0.859300
F1	0.910778	0.910969	0.913214	0.917684	0.915832	0.917991	0.913031	0.916270	0.911878

Conclusion

The scores achieved by many models in this project were relatively successful with the data provided. The SVM classifier after feature analysis and tuning performed the best as mentioned in the results section followed by the random forest and gradient boosting classifiers. Areas for improvement are varied but mainly pertain to the data as being representative of real life. The data provided was biased in a few notable ways. There was a heavy influence from the data being skewed toward representing white males in the US. This is remedied by collecting more diverse data to better represent the target cohort or rephrasing the prediction to specify the group most represented by the data.

Apart from the biases in the data the only other areas for improvement could potentially be methods of feature engineering to find new correlations in the available data and running the models on more powerful machines to cut down on training time. There were multiple hours spent just on training. Random forest or gradient boosting could be used to improve compute time over the SMV classifier as the performances are similar. This was a relatively cut-and-dry project that yielded many models that performed well.