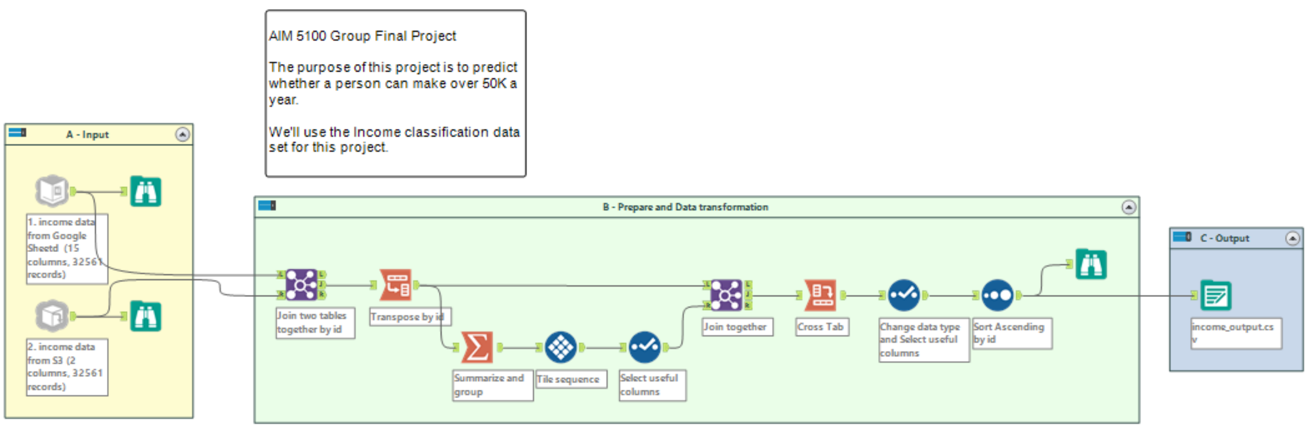
AIM 5100 Group Final Project

-by Manling Yang, Qi Sun, Xiaojia He

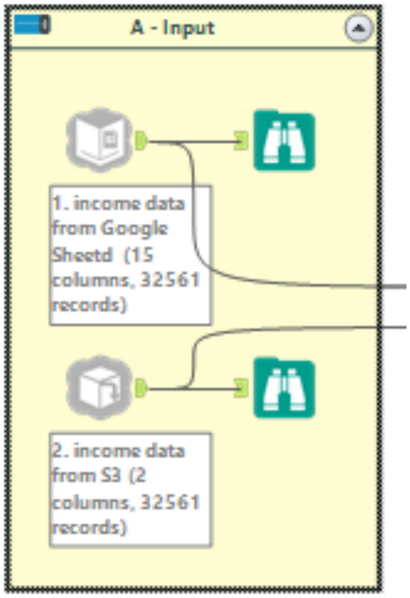
The purpose of this project is to predict whether a person can make over 50K a year. We'll use the Income classification data set for this project.

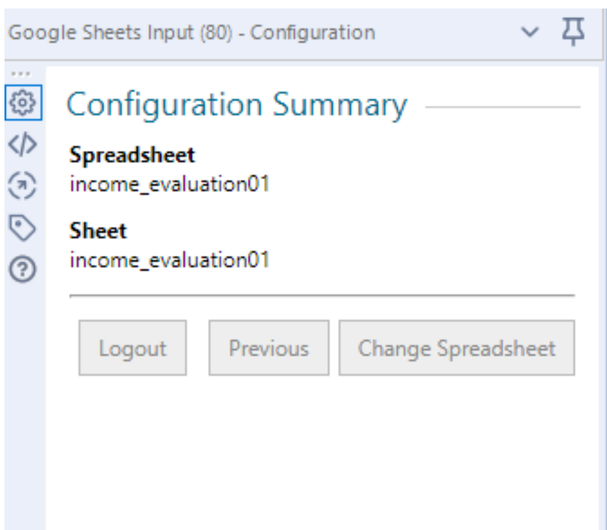
First of all, we use Alteryx to prepare data. We uploaded data from two data sources. One is from AWS S3 bucket, and the other one is from Google sheets. Then, we joined two datasets, encoded categorical variables, changed data type, and replaced missing values. Next, we output the cleaned data for building models by using DataRobot.

Below is a screenshot of the workflow on Alteryx.

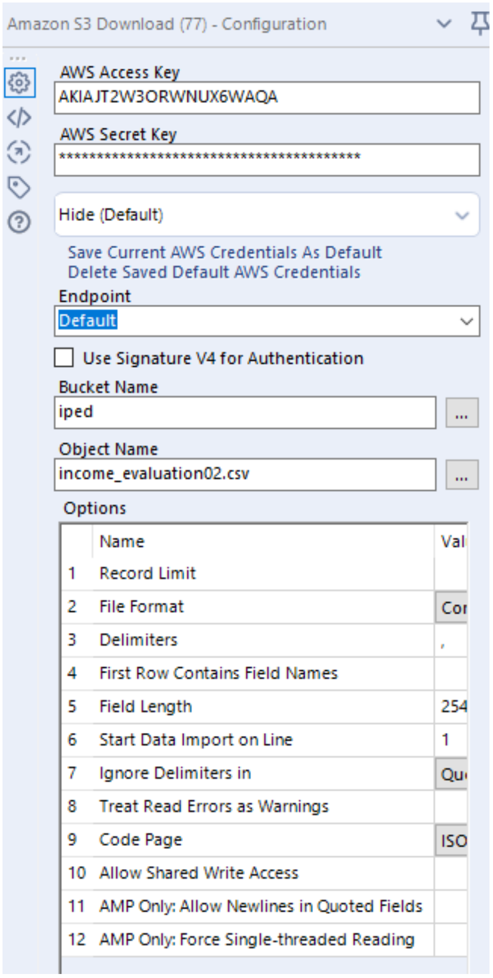


The following is the setup for uploading data from Google sheet.

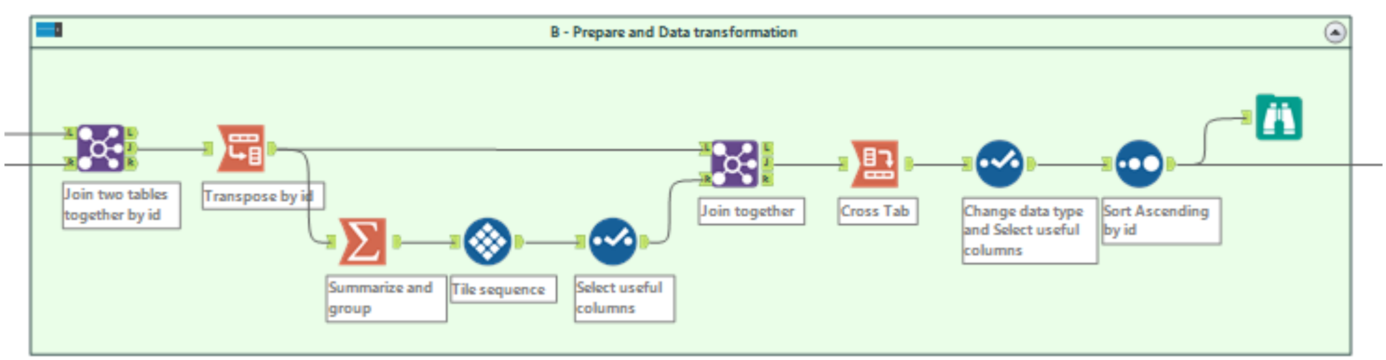




The following is the setup for uploading data from AWS S3.



Next, we performed data transform, including join tables, encode categorical variables, replace missing values, and change data type.



The followings are the screenshots for each step:

1. View Dataset 01:

Results - Google Sheets Input (80) - Output

15 of 15 Fields | Cell Viewer | * 4,103 of 32,561 records displayed(partial results) | Search | Data | Metadata

Record	id	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
1	1	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States
2	2	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States
3	3	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States
4	4	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States
5	5	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba
6	6	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States
7	7	49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica
8	8	52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	45	United-States
9	9	31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0	50	United-States
10	10	42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	5178	0	40	United-States
11	11	37	Private	280464	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	Black	Male	0	0	80	United-States

2. View Dataset 02:

Results - Amazon S3 Download (77) - Output

2 of 2 Fields | Cell Viewer | 32,561 records displayed

Record	id	income
1	1	<=50K
2	2	<=50K
3	3	<=50K
4	4	<=50K
5	5	<=50K
6	6	<=50K
7	7	<=50K
8	8	>50K
9	9	>50K
10	10	>50K
11	11	>50K

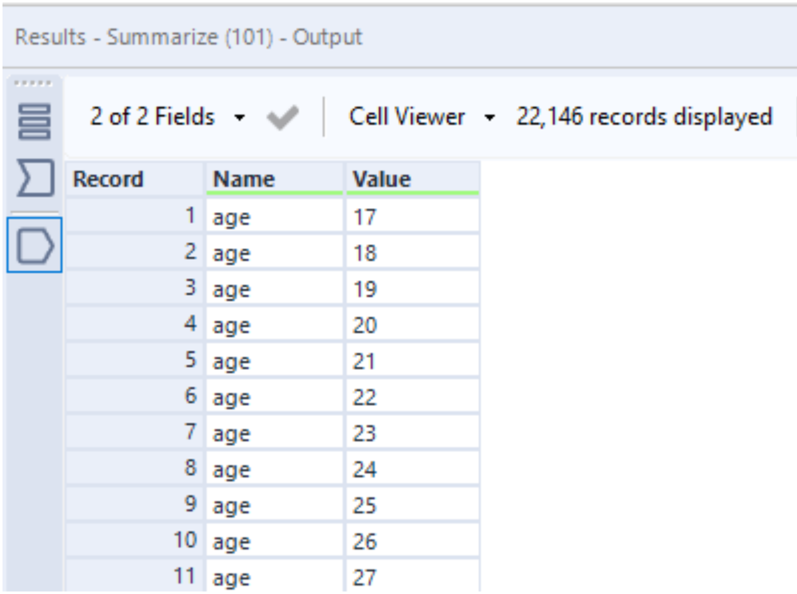
The most challenging part is to encode all categorical variables. After joining these two tables together, I transpose the table by id.

Results - Transpose (100) - Output

3 of 3 Fields | Cell Viewer | * 20,669 of 488,415 records displayed(partial results)

Record	id	Name	Value
1	1	age	39
2	1	workclass	State-gov
3	1	fnlwgt	77516
4	1	education	Bachelors
5	1	education-num	13
6	1	marital-status	Never-married
7	1	occupation	Adm-clerical
8	1	relationship	Not-in-family
9	1	race	White
10	1	sex	Male
11	1	capital-gain	2174

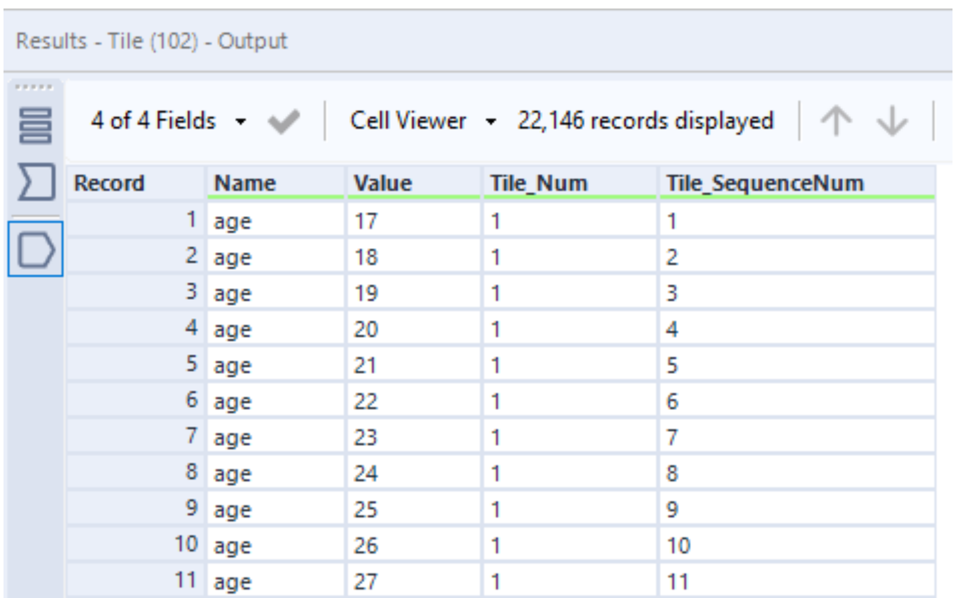
Then, we performed Summarize, Tile, Select, Join, and Cross tab, we got a table with all numerical variables. Below are the screenshots of the results from these steps:



Results - Summarize (101) - Output

2 of 2 Fields | Cell Viewer | 22,146 records displayed

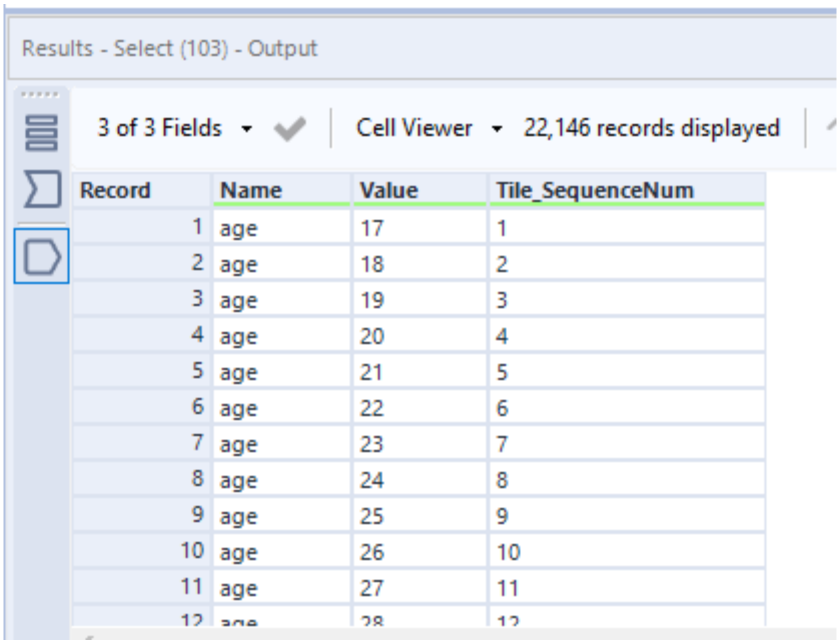
Record	Name	Value
1	age	17
2	age	18
3	age	19
4	age	20
5	age	21
6	age	22
7	age	23
8	age	24
9	age	25
10	age	26
11	age	27



Results - Tile (102) - Output

4 of 4 Fields | Cell Viewer | 22,146 records displayed

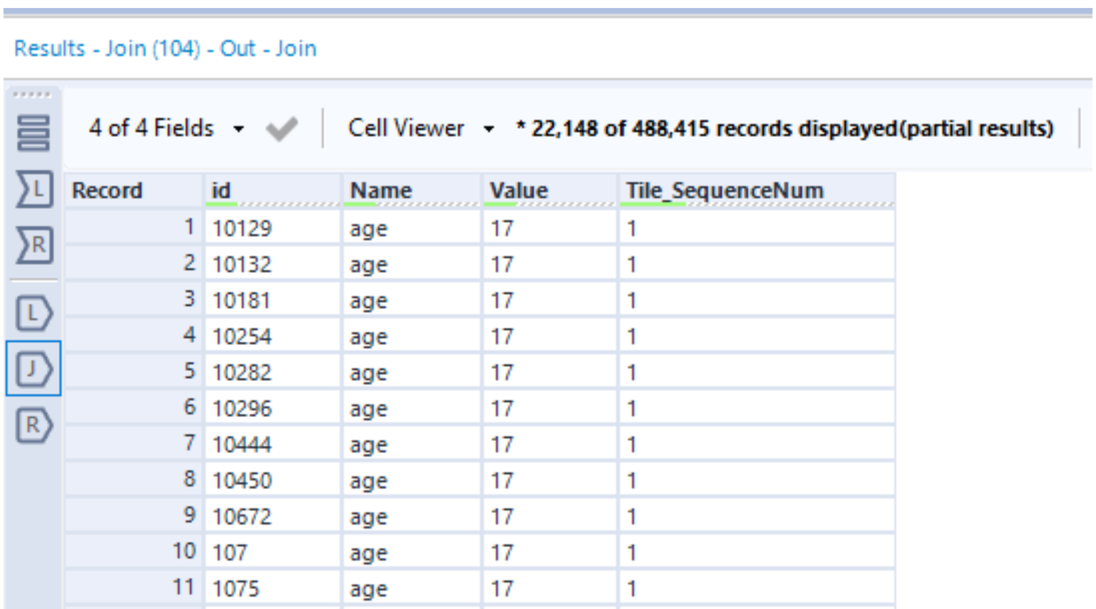
Record	Name	Value	Tile_Num	Tile_SequenceNum
1	age	17	1	1
2	age	18	1	2
3	age	19	1	3
4	age	20	1	4
5	age	21	1	5
6	age	22	1	6
7	age	23	1	7
8	age	24	1	8
9	age	25	1	9
10	age	26	1	10
11	age	27	1	11



Results - Select (103) - Output

3 of 3 Fields | Cell Viewer | 22,146 records displayed

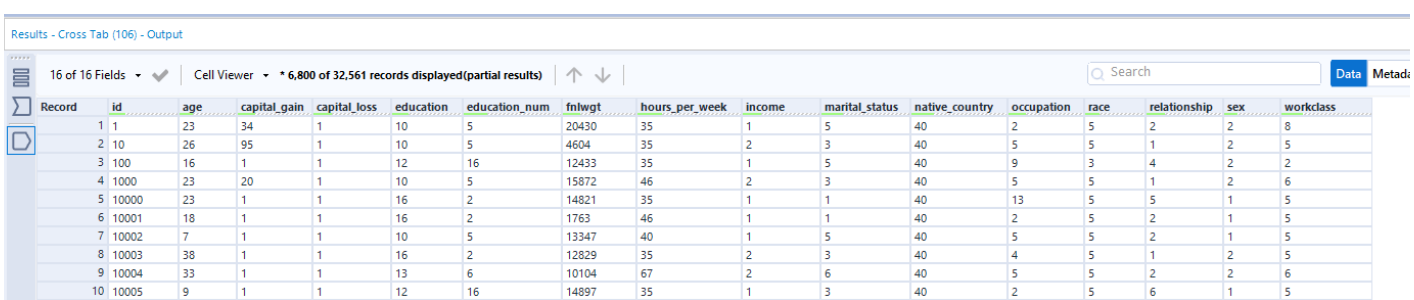
Record	Name	Value	Tile_SequenceNum
1	age	17	1
2	age	18	2
3	age	19	3
4	age	20	4
5	age	21	5
6	age	22	6
7	age	23	7
8	age	24	8
9	age	25	9
10	age	26	10
11	age	27	11
12	age	28	12



Results - Join (104) - Out - Join

4 of 4 Fields | Cell Viewer | * 22,148 of 488,415 records displayed(partial results)

Record	id	Name	Value	Tile_SequenceNum
1	10129	age	17	1
2	10132	age	17	1
3	10181	age	17	1
4	10254	age	17	1
5	10282	age	17	1
6	10296	age	17	1
7	10444	age	17	1
8	10450	age	17	1
9	10672	age	17	1
10	107	age	17	1
11	1075	age	17	1

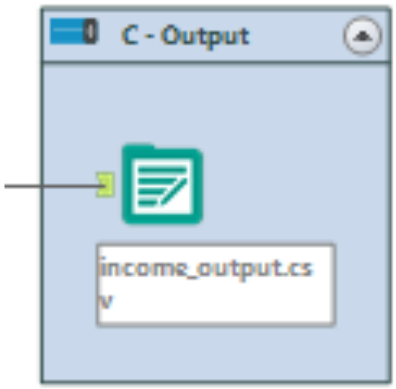


Results - Cross Tab (106) - Output

16 of 16 Fields | Cell Viewer | * 6,800 of 32,561 records displayed(partial results) | Search | Data | Metada

Record	id	age	capital_gain	capital_loss	education	education_num	fnlwgt	hours_per_week	income	marital_status	native_country	occupation	race	relationship	sex	workclass
1	1	23	34	1	10	5	20430	35	1	5	40	2	5	2	2	8
2	10	26	95	1	10	5	4604	35	2	3	40	5	5	1	2	5
3	100	16	1	1	12	16	12433	35	1	5	40	9	3	4	2	2
4	1000	23	20	1	10	5	15872	46	2	3	40	5	5	1	2	6
5	10000	23	1	1	16	2	14821	35	1	1	40	13	5	5	1	5
6	10001	18	1	1	16	2	1763	46	1	1	40	2	5	2	1	5
7	10002	7	1	1	10	5	13347	40	1	5	40	5	5	2	1	5
8	10003	38	1	1	16	2	12829	35	2	3	40	4	5	1	2	5
9	10004	33	1	1	13	6	10104	67	2	6	40	5	5	2	2	6
10	10005	9	1	1	12	16	14897	35	1	3	40	2	5	6	1	5
11	10006	35	1	1	8	4	2110	46	1	3	40	13	5	1	2	5

Lastly, we output the cleaned and transformed data for the use of DataRobot.



DataRobot:

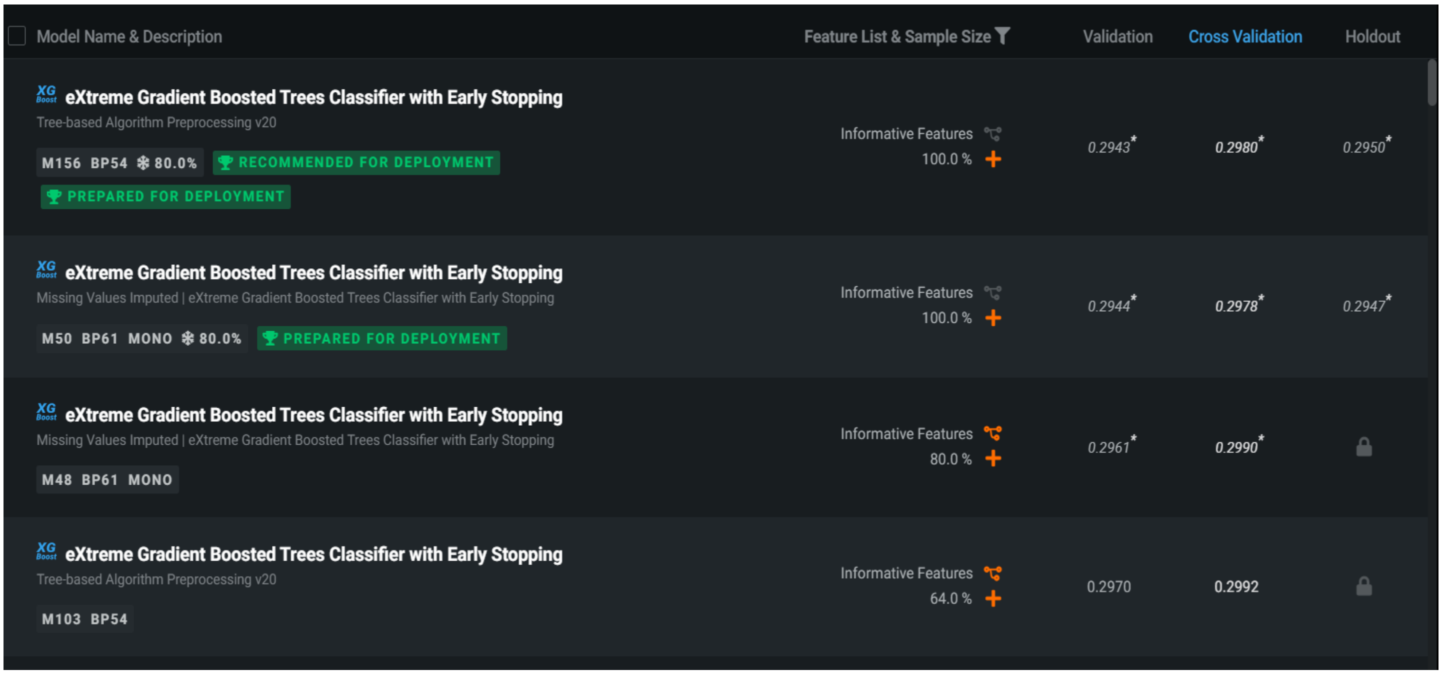
1. Select all features to create the model.

Menu | Search | Feature List: All Features | View Raw Data | + Create feature list | 1-16 of 16

Feature Name	Data Quality	Index	Importance	Var Type	Unique	Missing	Mean	Std Dev	Median	Min	Max
income TARGET		9	Target	Numeric	2	0	1.24	0.43	1	1	2
marital_status		10		Numeric	7	0	3.61	1.51	3	1	7
relationship		14		Numeric	6	0	2.45	1.61	2	1	6
age		2		Numeric	72	0	22.56	13.61	21	1	73
education_num		6		Numeric	16	0	8.86	5.87	6	1	16
occupation		12		Numeric	15	0	7.56	4.22	8	1	15
hours_per_week		8		Numeric	92	0	36.05	13.04	35	1	94
education		5		Numeric	16	0	11.28	3.89	12	1	16
capital_gain	i	3		Numeric	119	0	6.69	21.92	1	1	119
sex		15		Numeric	2	0	1.67	0.47	2	1	2
capital_loss	i	4		Numeric	90	0	2.81	9.05	1	1	92
race		13		Numeric	5	0	4.67	0.85	5	1	5

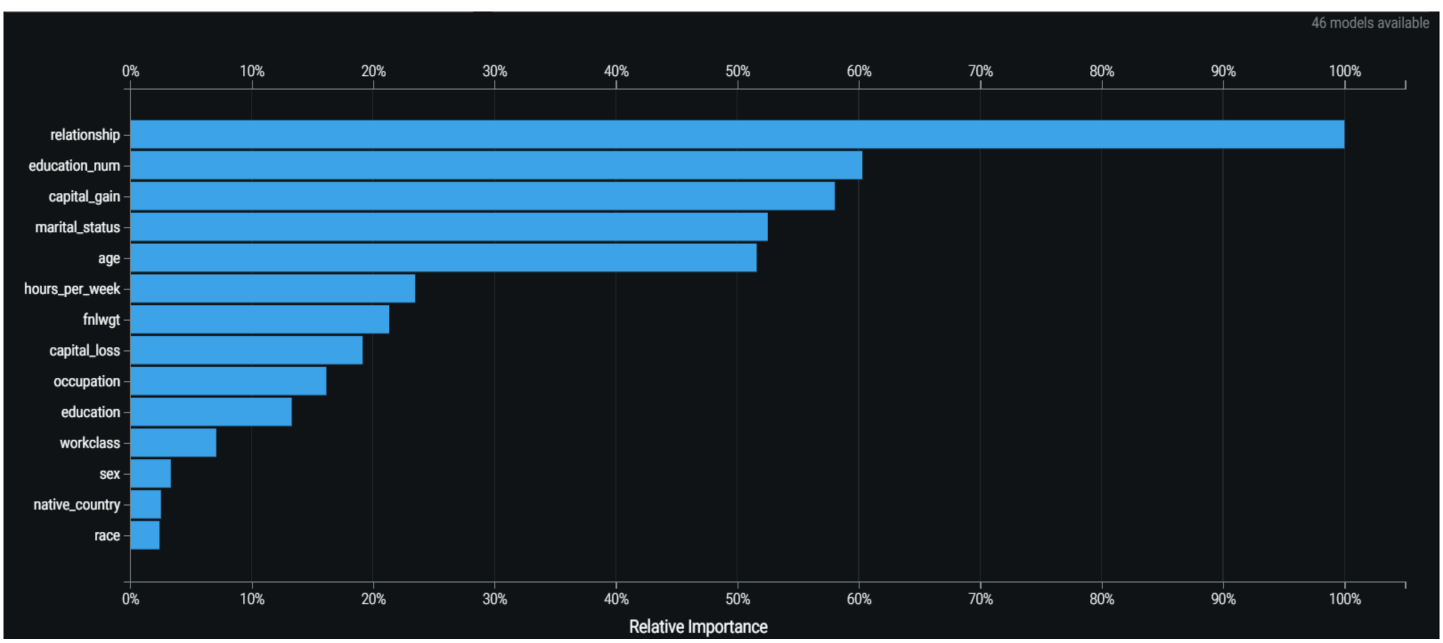
2. Gain the best model: eXtreme Gradient Boosted Tree Classifier with Early Stopping

Datarobot generates 73 models. We find that the Gradient Boosted Tree Classifier models are the better models that have higher AUC scores and lower RMSE scores. And the DataRobot recommended eXtreme Gradient Boosted Tree Classifier with Early Stopping with M156 BP54 are the best model to deploy.

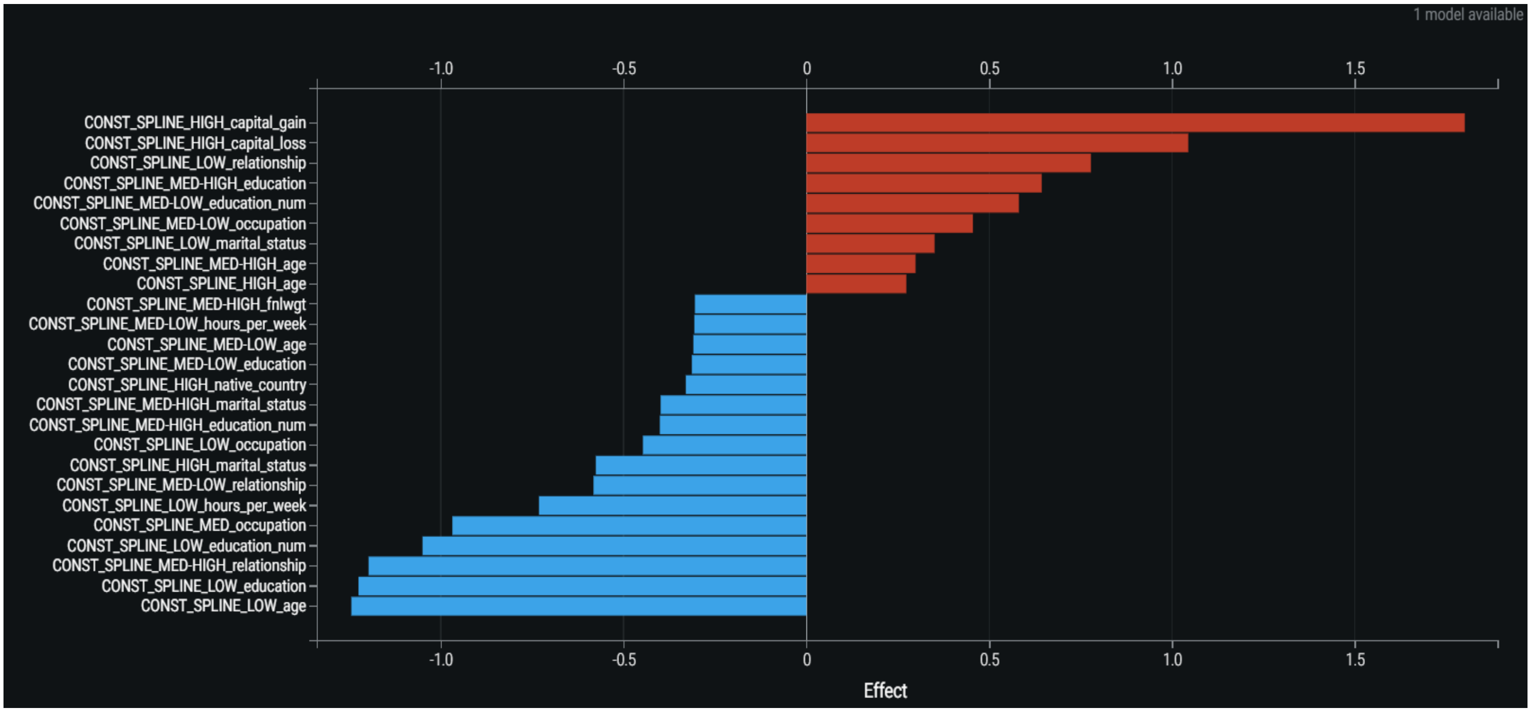


3. Insight of the best model: Three base variable importance

Relationship, education_num, capital_gain, marital_status, and age are the top five features that have strong impacts on the models.

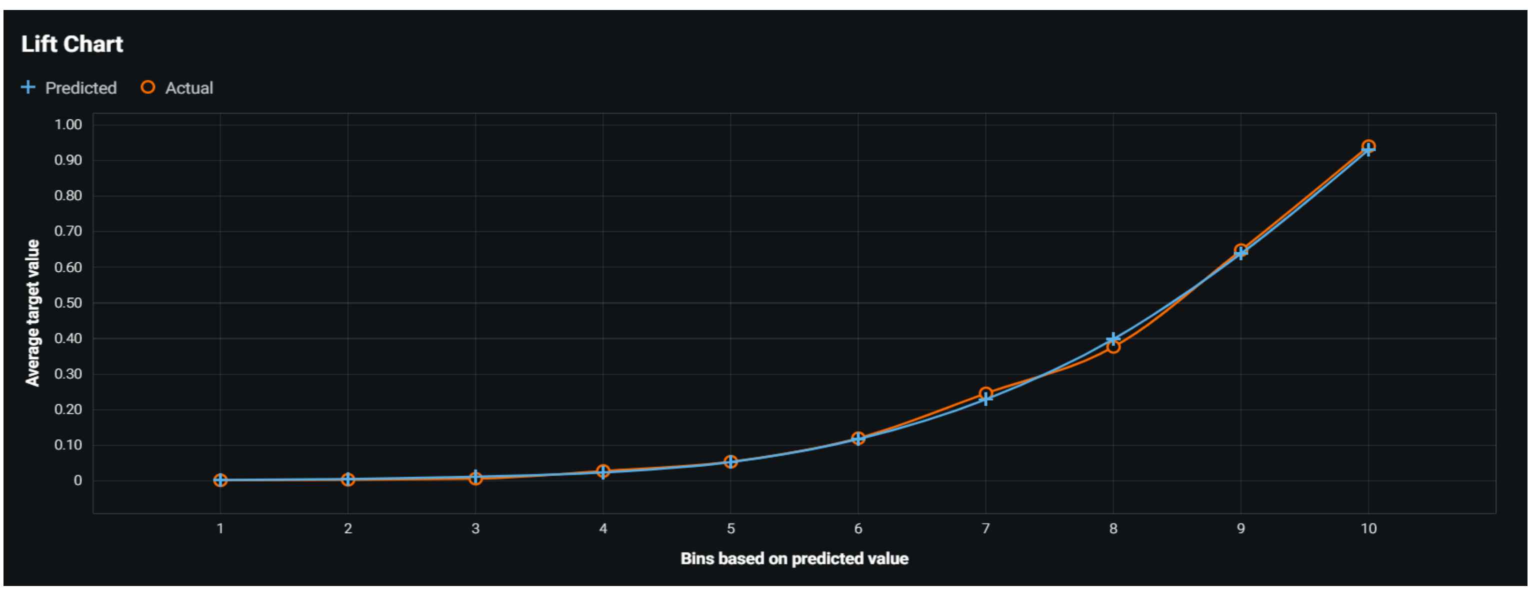


4. Insight of the best model: variable effect



5. Live Chart: the difference between prediction and actual

The prediction is almost the same as the actual results. It means the model is great.



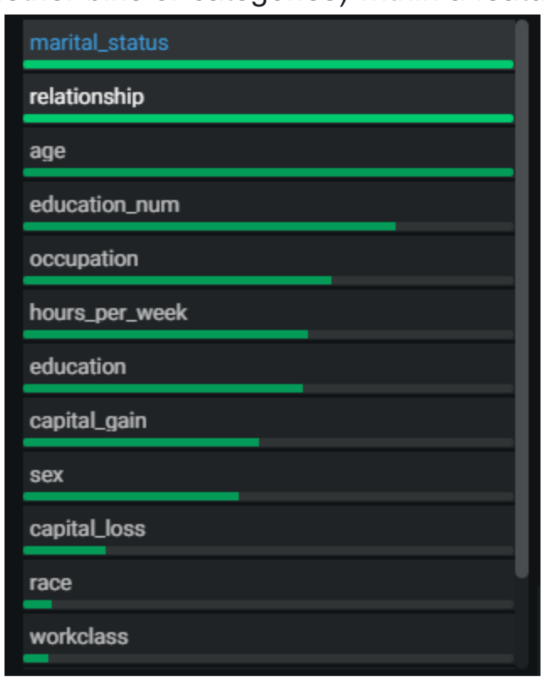
6. Selection summary: the detail of F1 scores and accuracy are high which means the model is good.

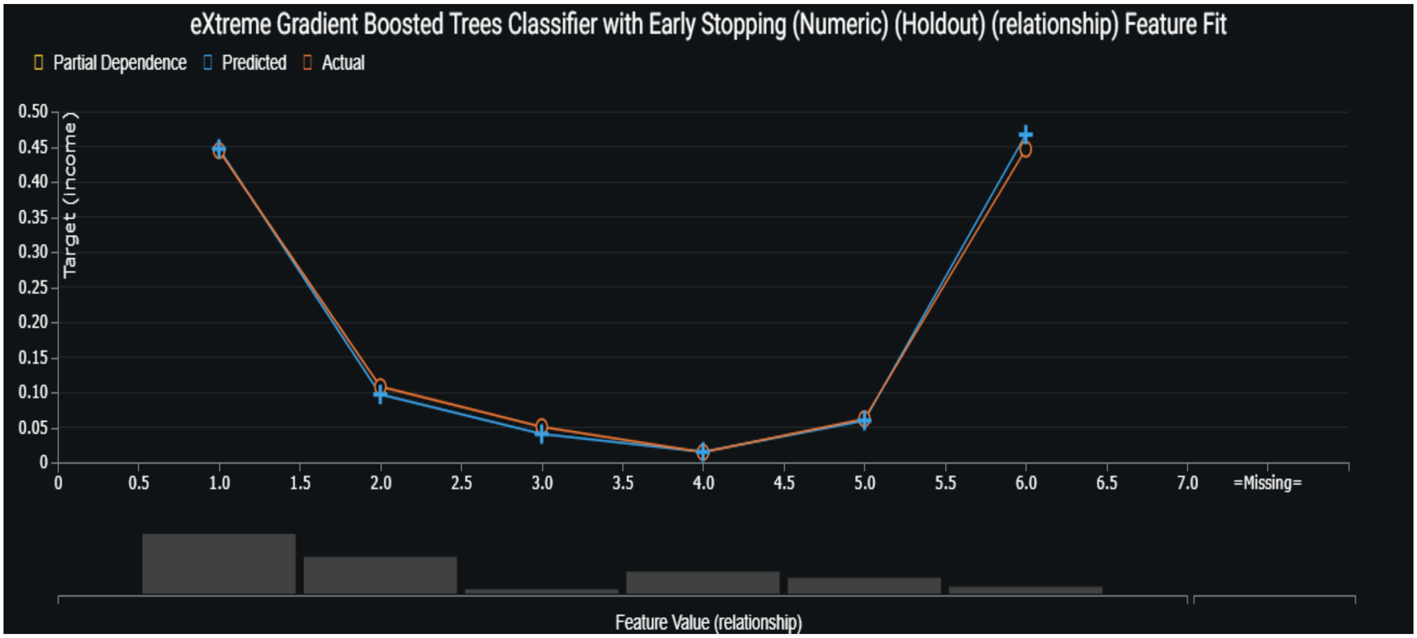
Selection Summary Export

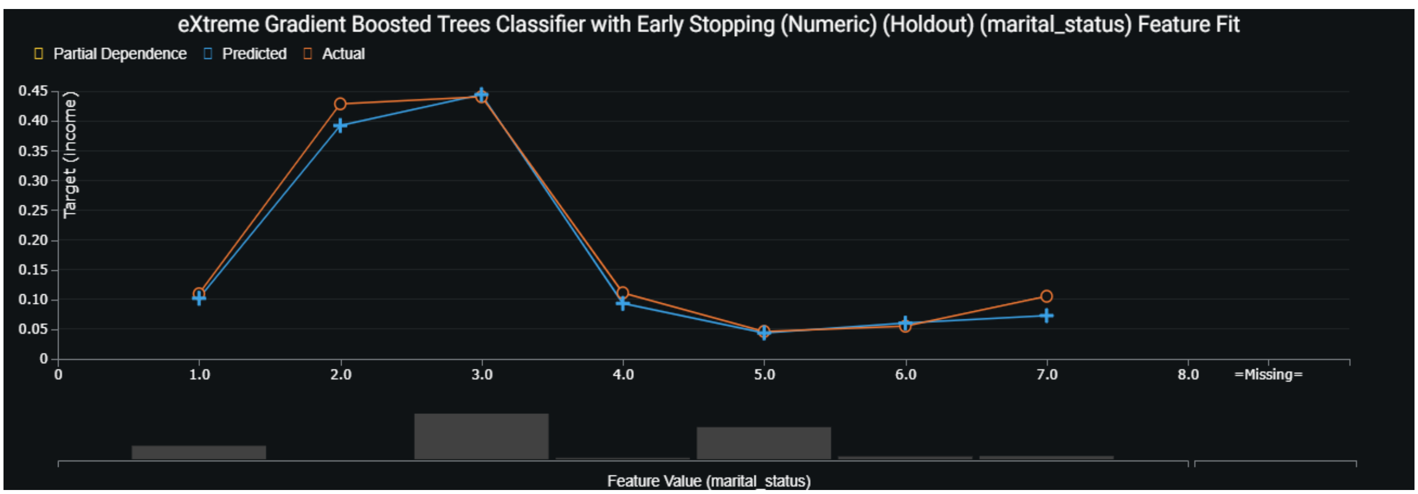
F1 Score	True Positive Rate (Sensitivity)	False Positive Rate (Fallout)	True Negative Rate (Specificity)	Positive Predictive Value (Precision)	Negative Predictive Value	Accuracy	Matthews Correlation Coefficient
0.7275	0.7623	0.1058	0.8942	0.6957	0.9222	0.8625	0.6369

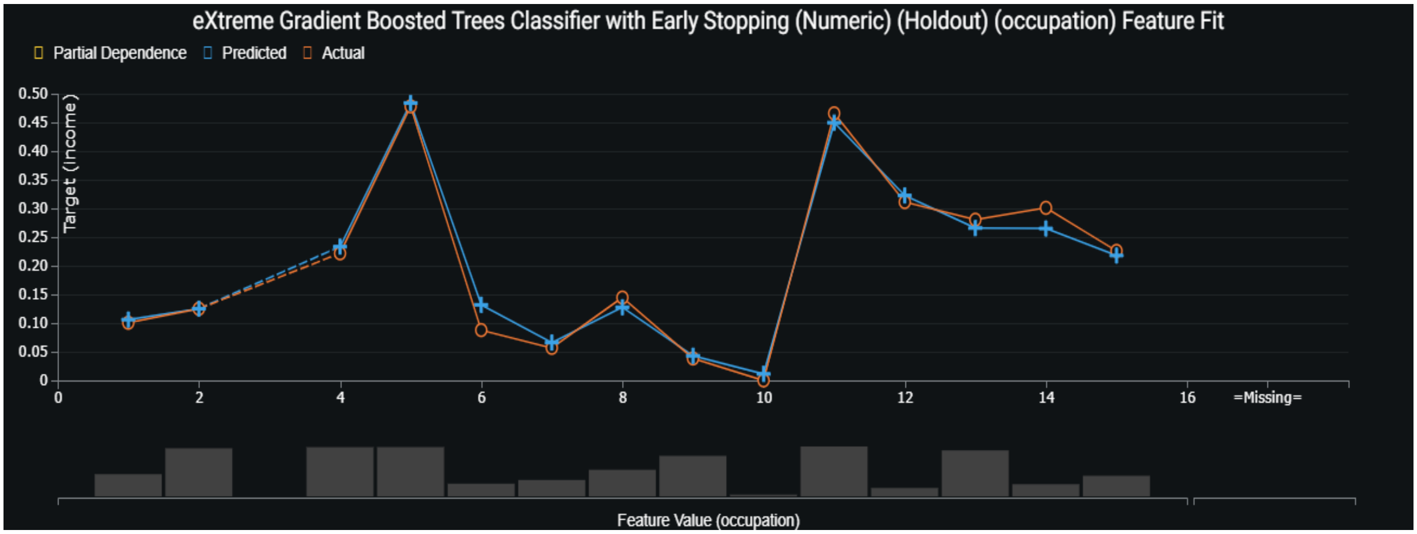
		Predicted -	Predicted +	
Actual	-	17684 (TN)	2092 (FP)	19776
Actual	+	1491 (FN)	4782 (TP)	6273
		19175	6874	26049

7. Summary of feature fit: we discovered marital_status, relationship, and age have the best feature fits. It means how well the model does when predicting a particular subset of data (whether bins or categories) within a feature

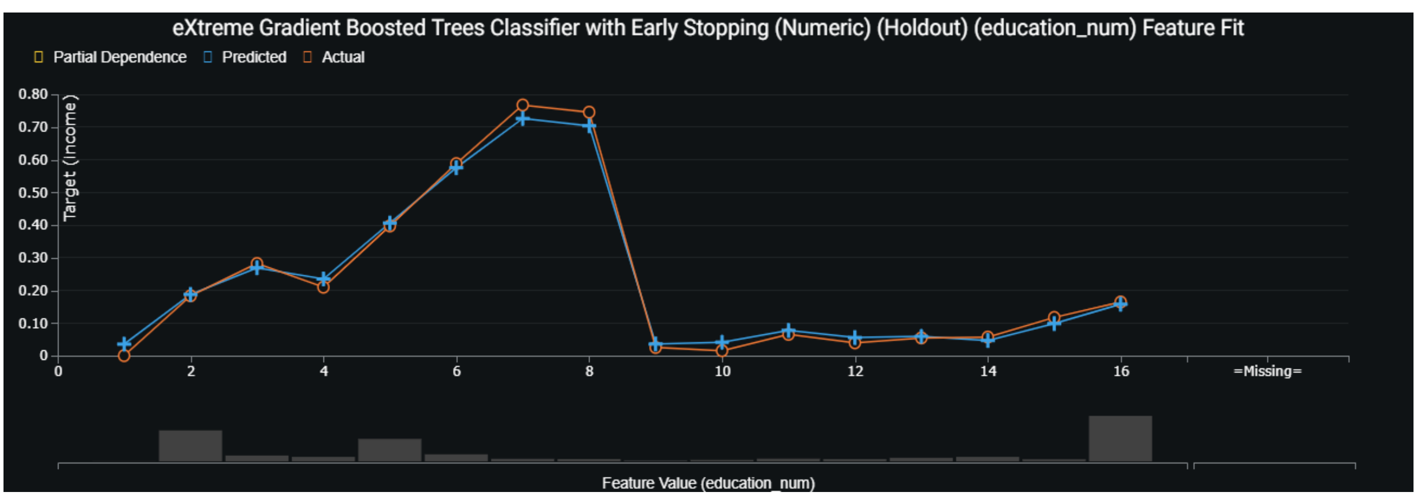



eXtreme Gradient Boosted Trees Classifier with Early Stopping (Numeric) (Holdout) (relationship) Feature Fit
Partial Dependence
Predicted
Actual
Target (income)
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0
0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
=Missing=
Feature Value (relationship)

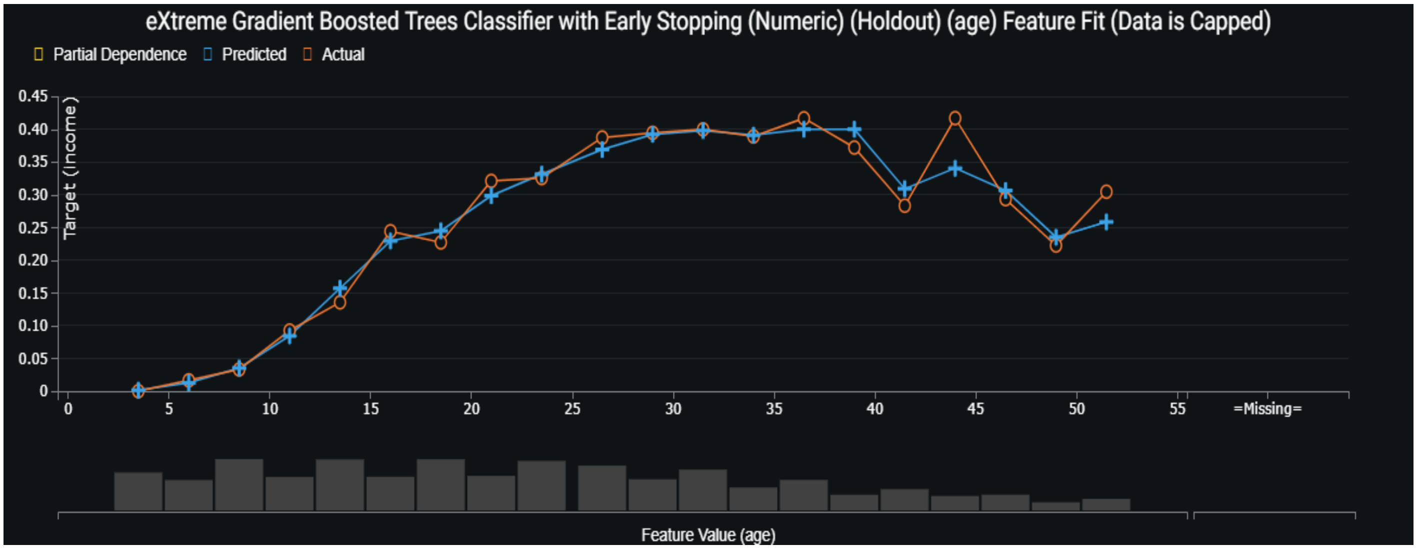

eXtreme Gradient Boosted Trees Classifier with Early Stopping (Numeric) (Holdout) (marital_status) Feature Fit
Partial Dependence
Predicted
Actual
Target (income)
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0
0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
=Missing=
Feature Value (marital_status)

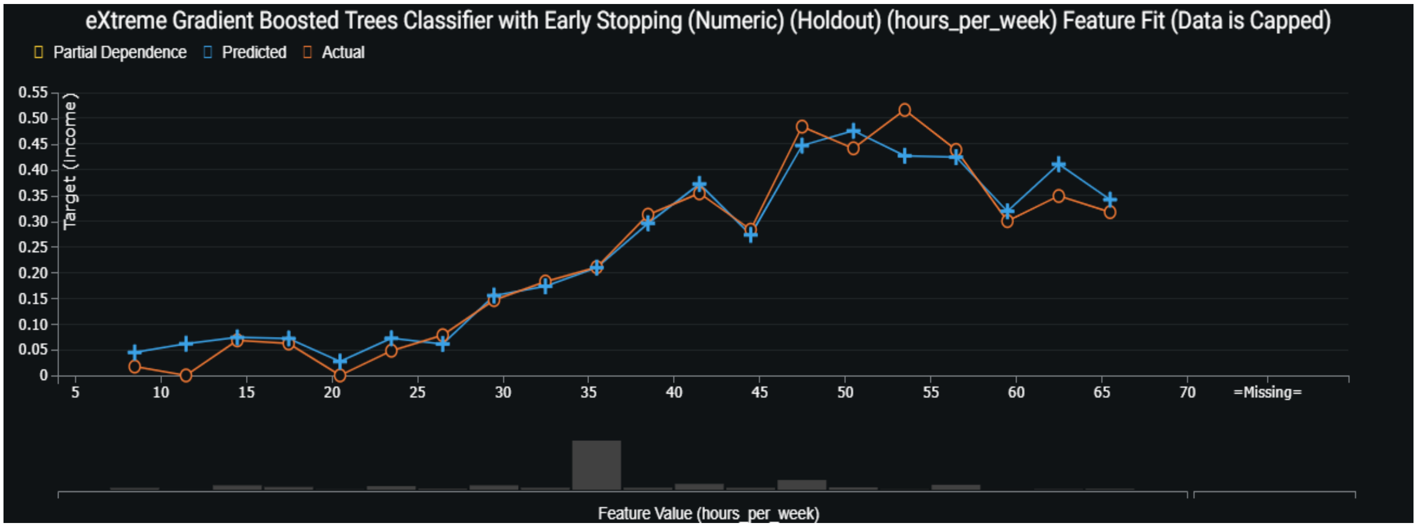


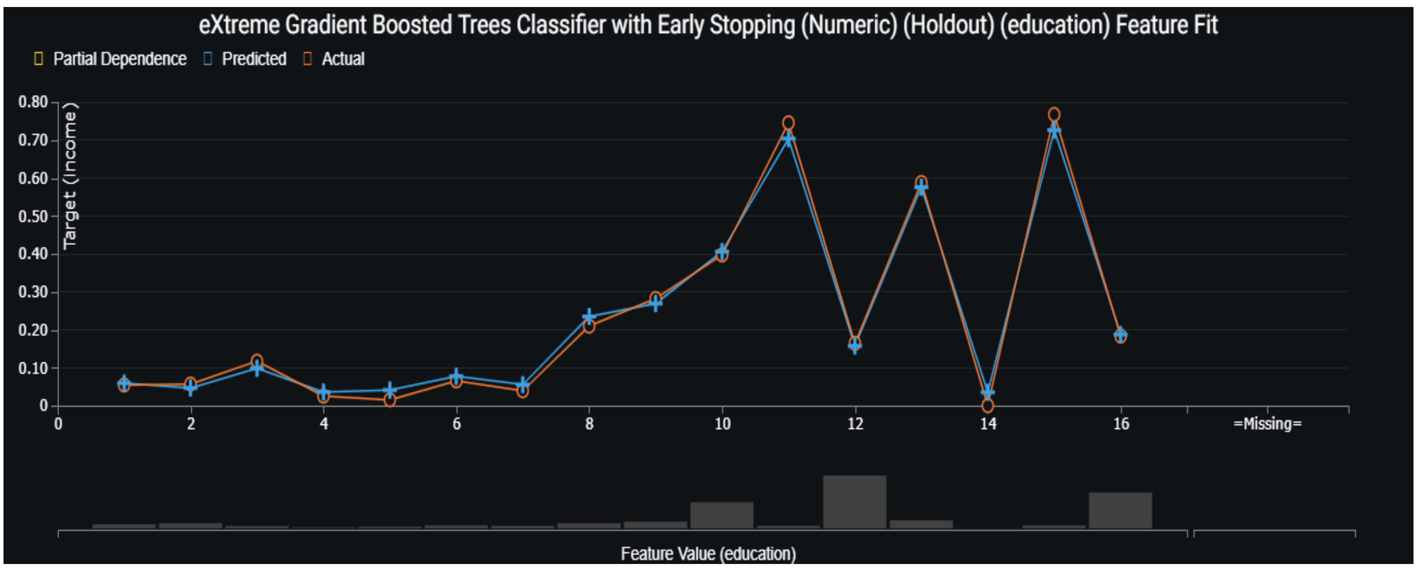
eXtreme Gradient Boosted Trees Classifier with Early Stopping (Numeric) (Holdout) (occupation) Feature Fit
Partial Dependence
Predicted
Actual
Target (income)
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0
0
2
4
6
8
10
12
14
16
=Missing=
Feature Value (occupation)

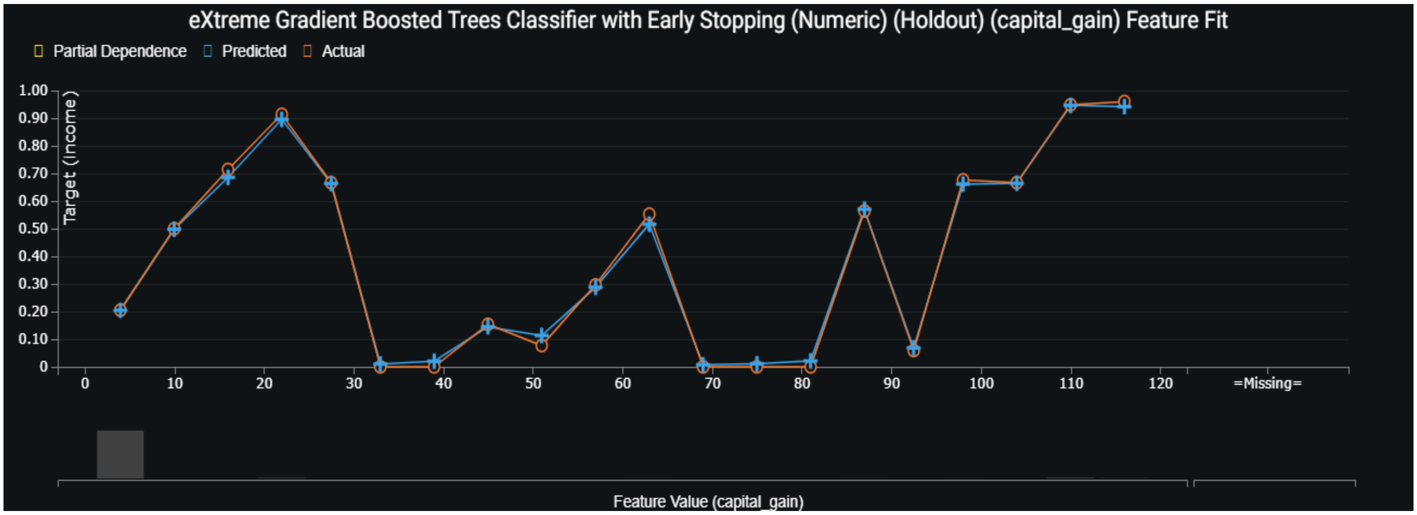


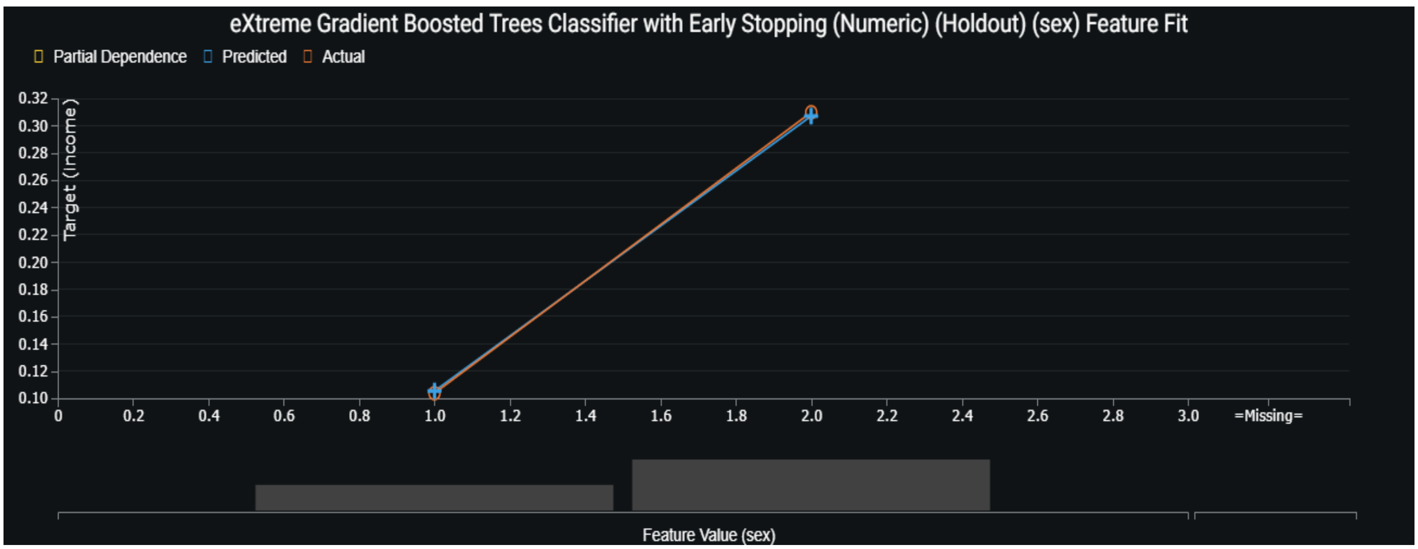
eXtreme Gradient Boosted Trees Classifier with Early Stopping (Numeric) (Holdout) (education_num) Feature Fit
Partial Dependence
Predicted
Actual
Target (income)
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0
0
2
4
6
8
10
12
14
16
=Missing=
Feature Value (education_num)

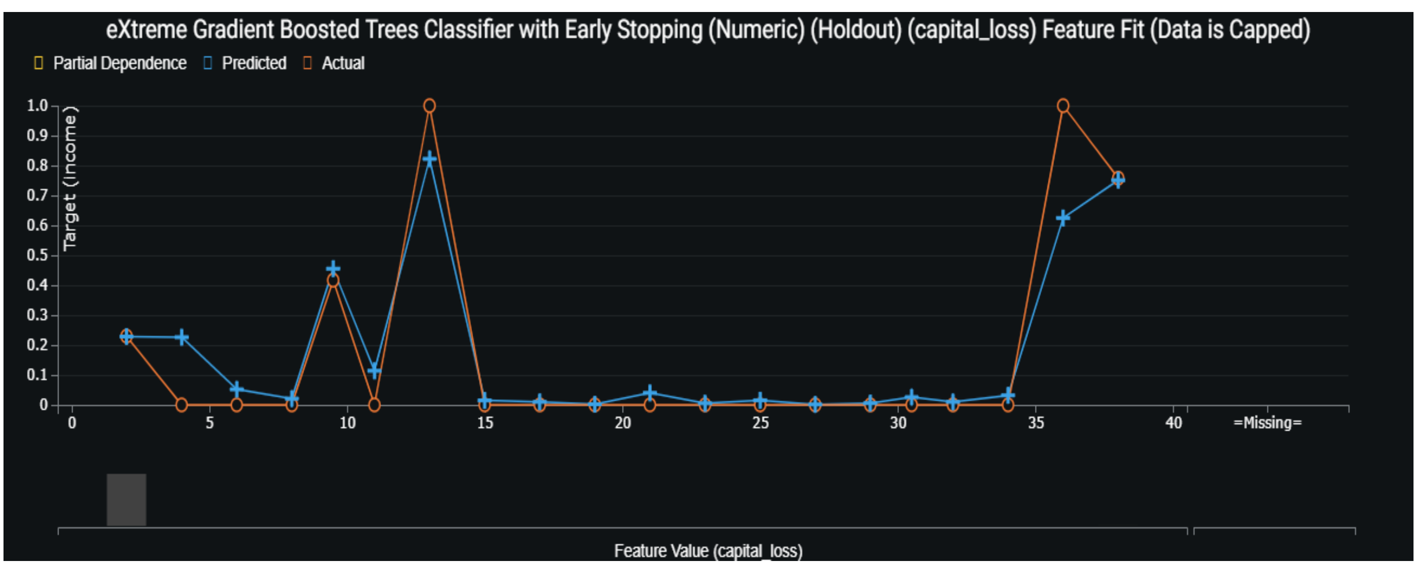

eXtreme Gradient Boosted Trees Classifier with Early Stopping (Numeric) (Holdout) (age) Feature Fit (Data is Capped)
Partial Dependence
Predicted
Actual
Target (income)
=Missing=
Feature Value (age)


eXtreme Gradient Boosted Trees Classifier with Early Stopping (Numeric) (Holdout) (hours_per_week) Feature Fit (Data is Capped)
Partial Dependence
Predicted
Actual
Target (income)
=Missing=
Feature Value (hours_per_week)

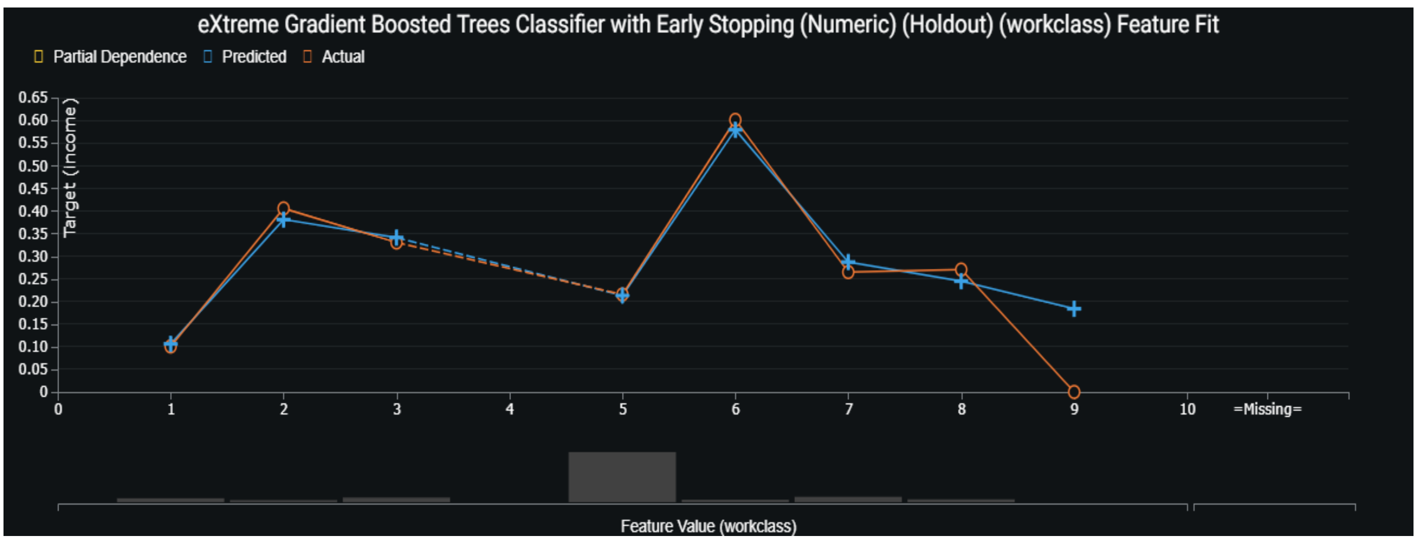

eXtreme Gradient Boosted Trees Classifier with Early Stopping (Numeric) (Holdout) (education) Feature Fit
Partial Dependence
Predicted
Actual
Target (income)
=Missing=
Feature Value (education)


eXtreme Gradient Boosted Trees Classifier with Early Stopping (Numeric) (Holdout) (capital_gain) Feature Fit
Partial Dependence
Predicted
Actual
Target (income)
Feature Value (capital_gain)
=Missing=

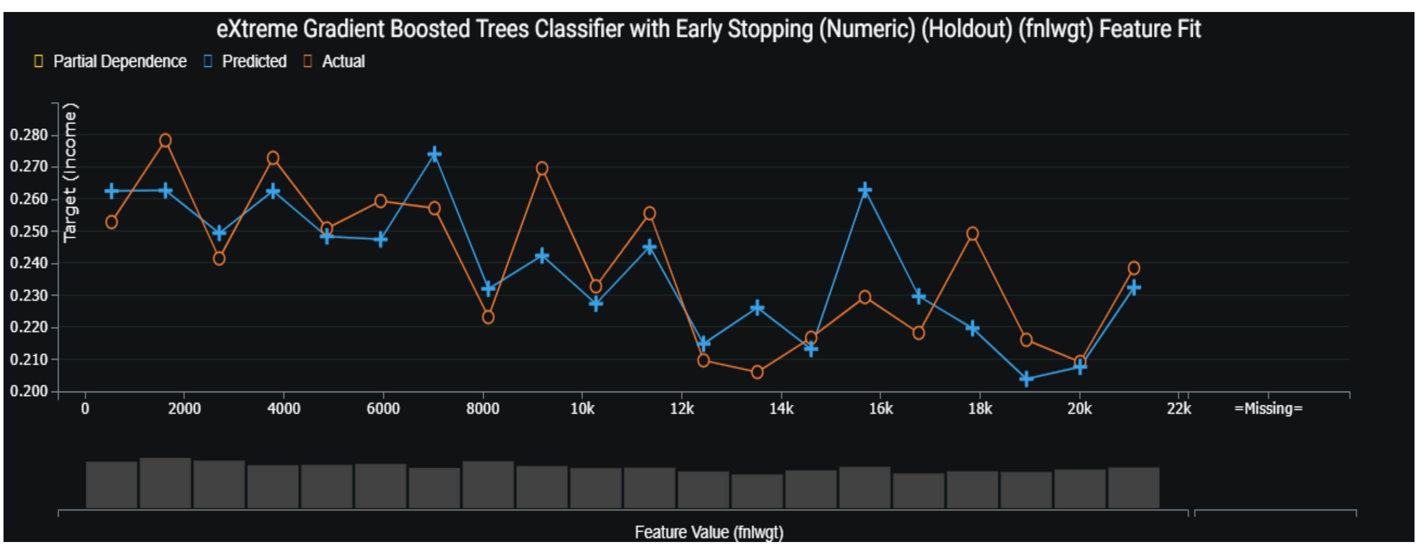

eXtreme Gradient Boosted Trees Classifier with Early Stopping (Numeric) (Holdout) (sex) Feature Fit
Partial Dependence
Predicted
Actual
Target (income)
Feature Value (sex)
=Missing=


eXtreme Gradient Boosted Trees Classifier with Early Stopping (Numeric) (Holdout) (capital_loss) Feature Fit (Data is Capped)
Partial Dependence
Predicted
Actual
Target (income)
Feature Value (capital_loss)
=Missing=

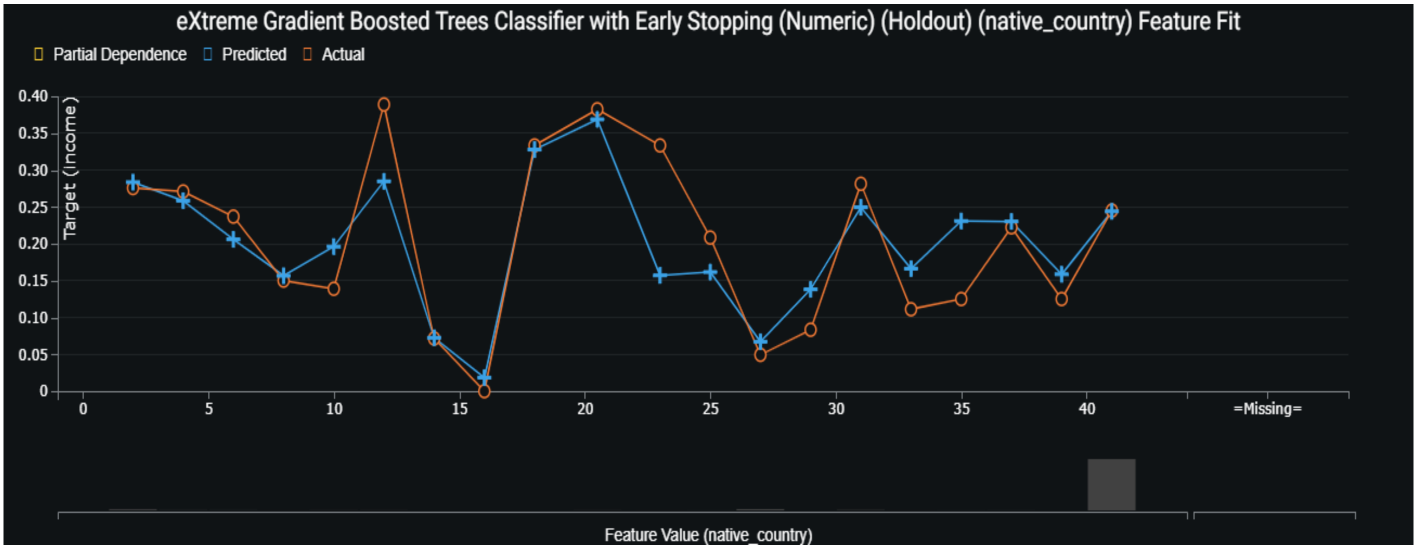
eXtreme Gradient Boosted Trees Classifier with Early Stopping (Numeric) (Holdout) (race) Feature Fit
Partial Dependence
Predicted
Actual
Target (income)
=Missing=
Feature Value (race)



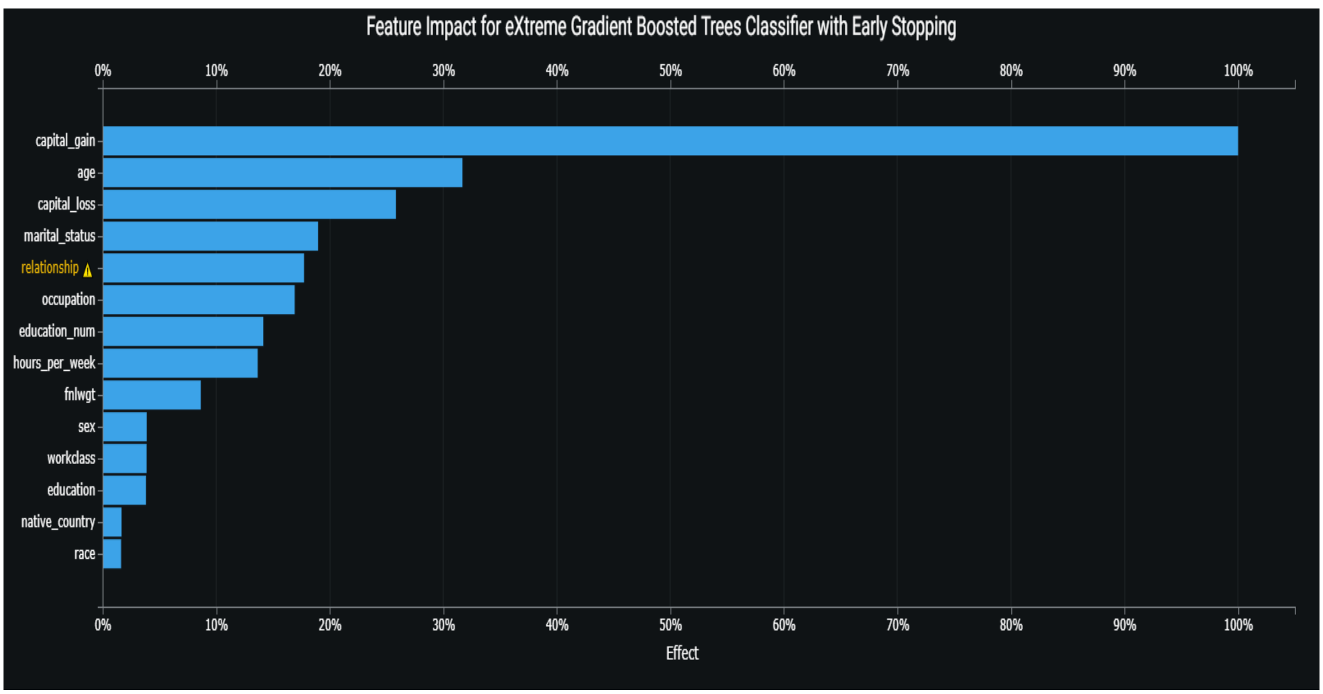
eXtreme Gradient Boosted Trees Classifier with Early Stopping (Numeric) (Holdout) (workclass) Feature Fit
Partial Dependence
Predicted
Actual
Target (income)
=Missing=
Feature Value (workclass)



eXtreme Gradient Boosted Trees Classifier with Early Stopping (Numeric) (Holdout) (fnlwgt) Feature Fit
Partial Dependence
Predicted
Actual
Target (income)
=Missing=
Feature Value (fnlwgt)



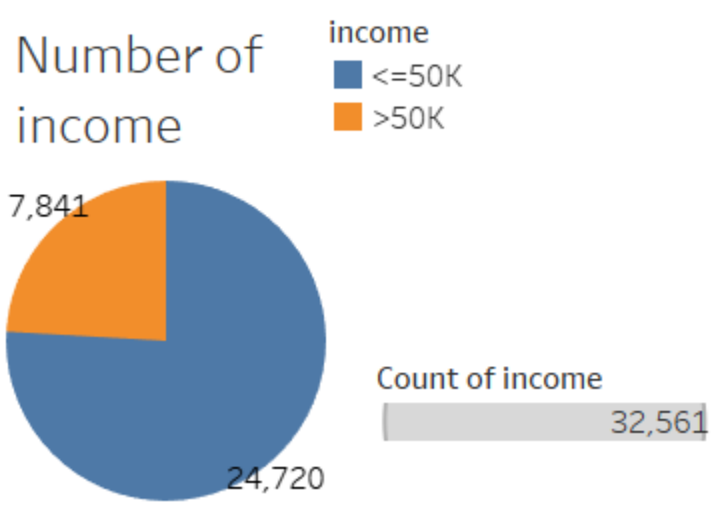
8. Feature Impact: The best model is extremely influenced by Capital_gain which is almost 100%. Age and capital_loss are the next and have around 20% impact.



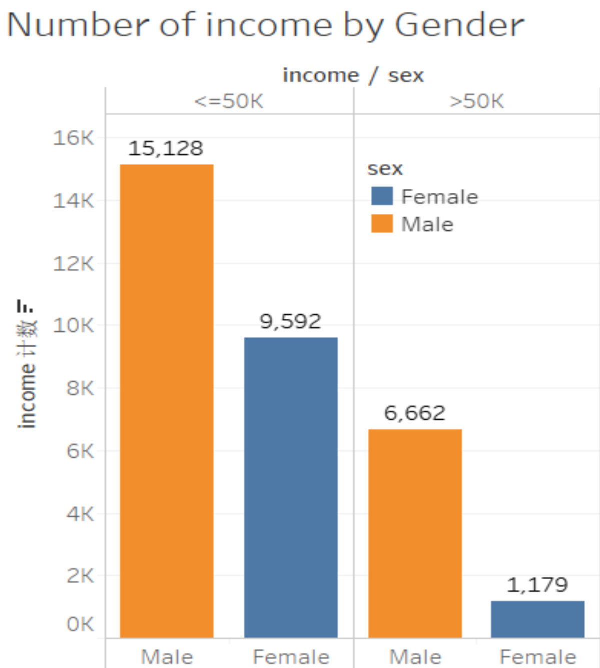
9. Example of prediction explanation

ID	PREDICTION	EXPLANATIONS		
4625	1.000	+++ capital_gain = 119	++ age = 31	++ relationship = 1
22252	1.000	+++ capital_gain = 111	++ age = 29	++ occupation = 5
13947	1.000	+++ capital_gain = 111	++ age = 29	++ occupation = 5
29982	0.000	--- age = 1	-- capital_loss = 18	-- hours_per_week = 2
7186	0.000	--- age = 3	--- capital_gain = 67	-- marital_status = 5
19236	0.000	--- age = 1	-- hours_per_week = 10	-- capital_loss = 18

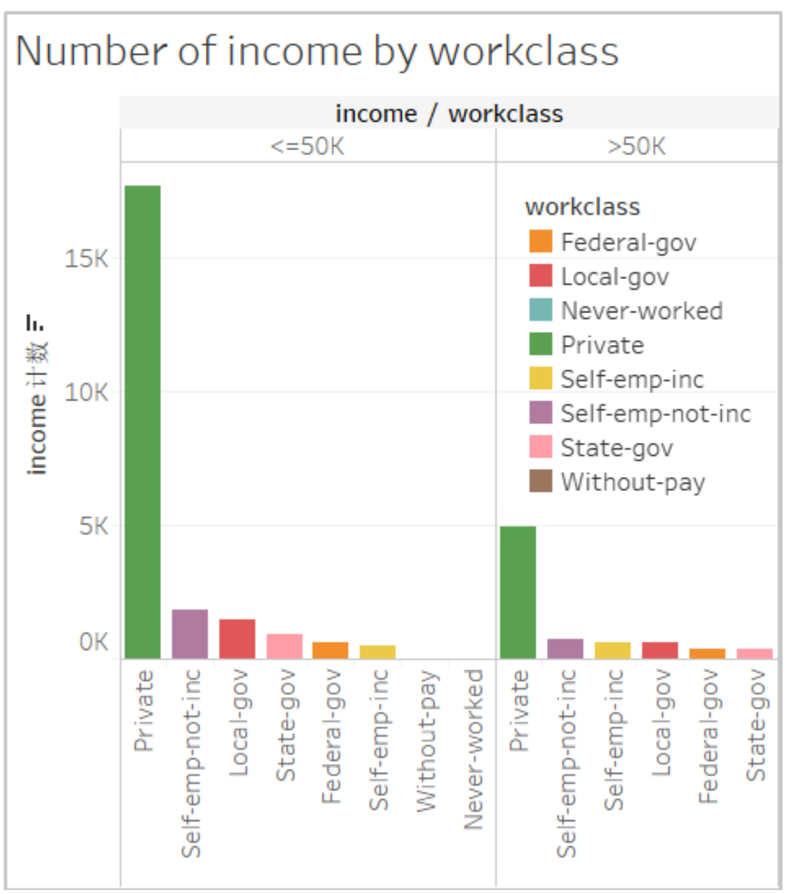
Tableau: the visualizations are the same as Python works.



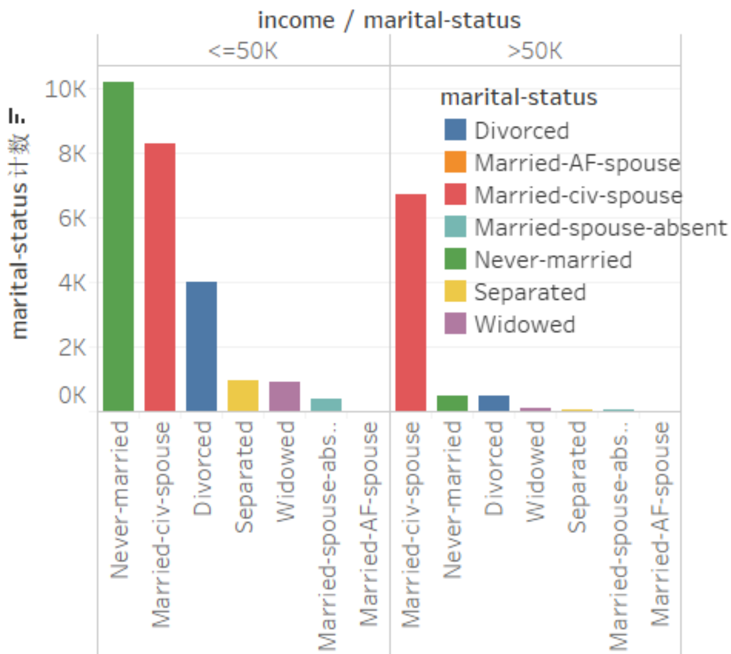
Dfdsfd


Number of income by Gender
income / sex
<=50K
>50K
16K
14K
12K
10K
8K
6K
4K
2K
0K
income 计数
15,128
9,592
6,662
1,179
sex
Female
Male
Male
Female
Male
Female

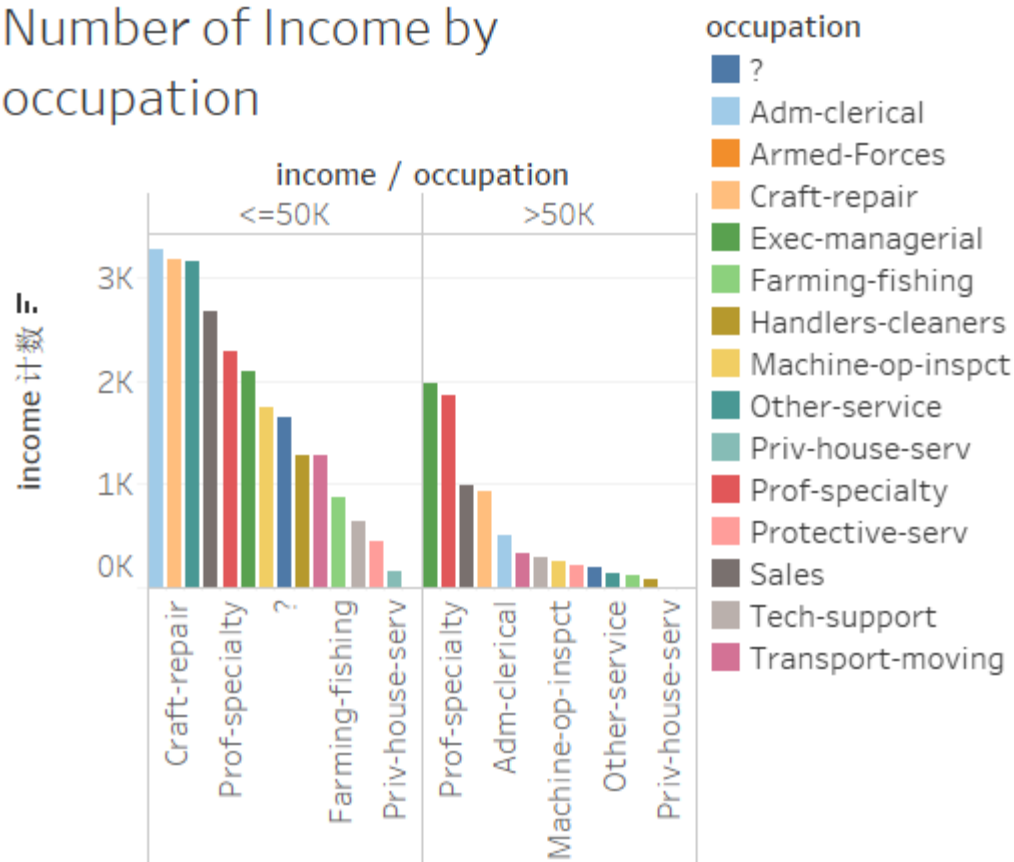
dfdfd


Number of income by workclass
income / workclass
<=50K
>50K
15K
10K
5K
0K
income 计数
workclass
Federal-gov
Local-gov
Never-worked
Private
Self-emp-inc
Self-emp-not-inc
State-gov
Without-pay
Private
Self-emp-not-inc
Local-gov
State-gov
Federal-gov
Self-emp-inc
Without-pay
Never-worked
Private
Self-emp-not-inc
Self-emp-inc
Local-gov
Federal-gov
State-gov

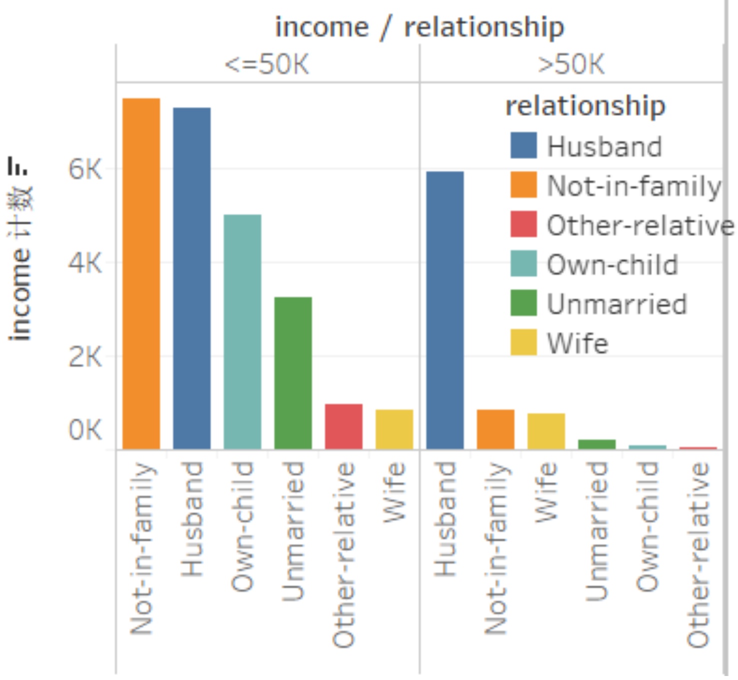
Number of income by Married status



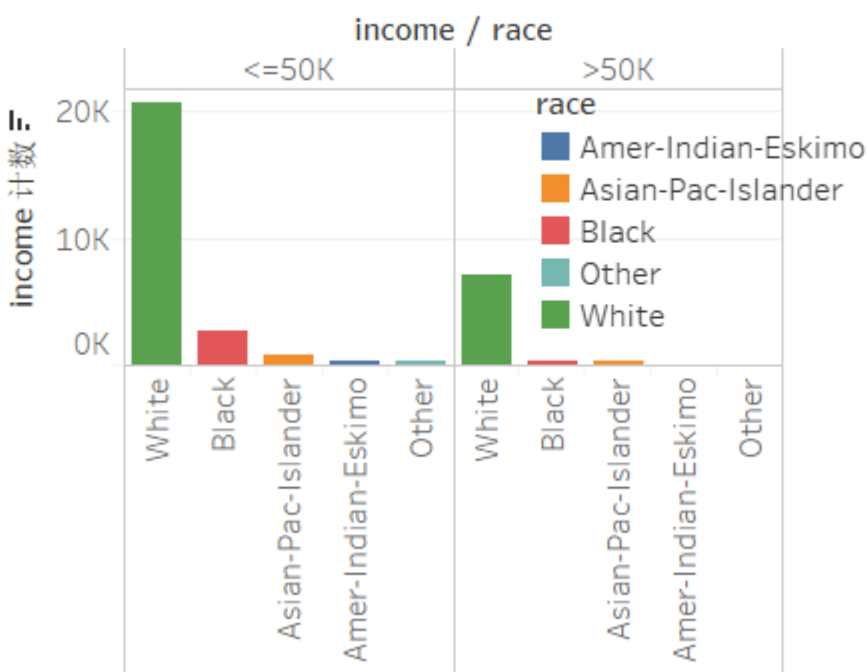
Number of Income by occupation

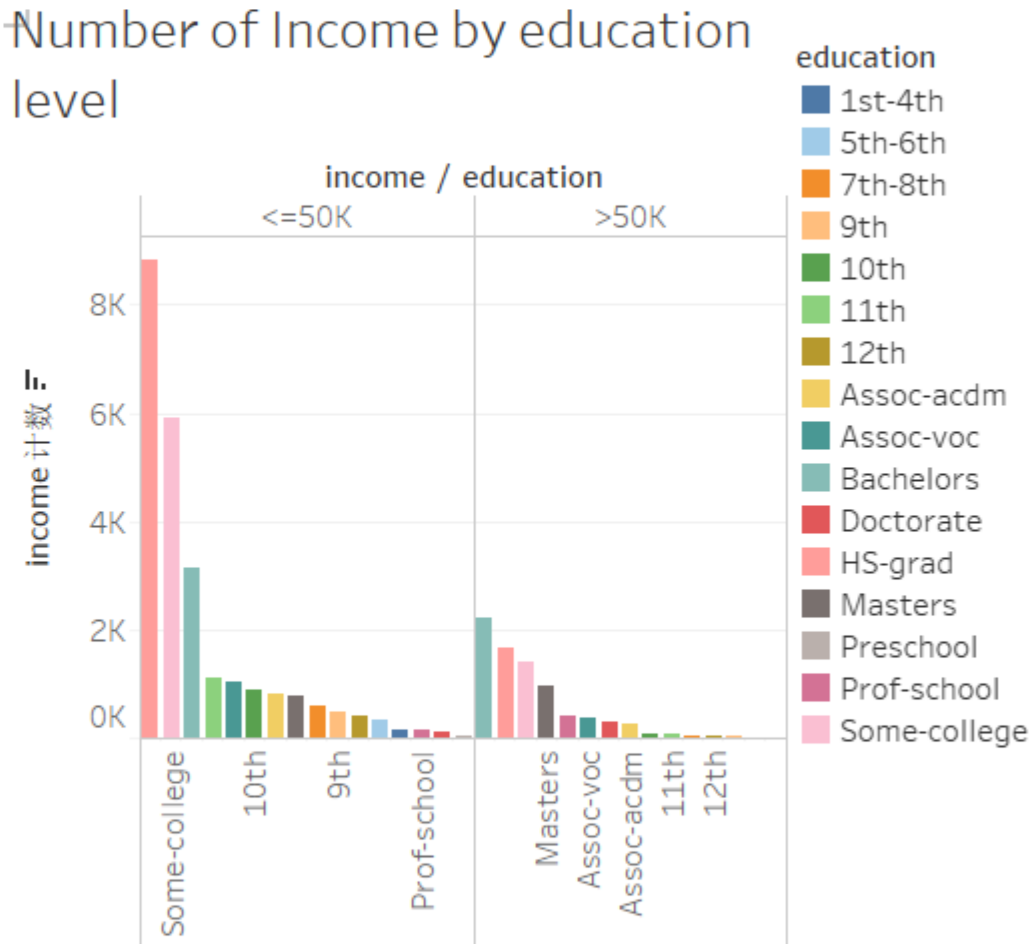


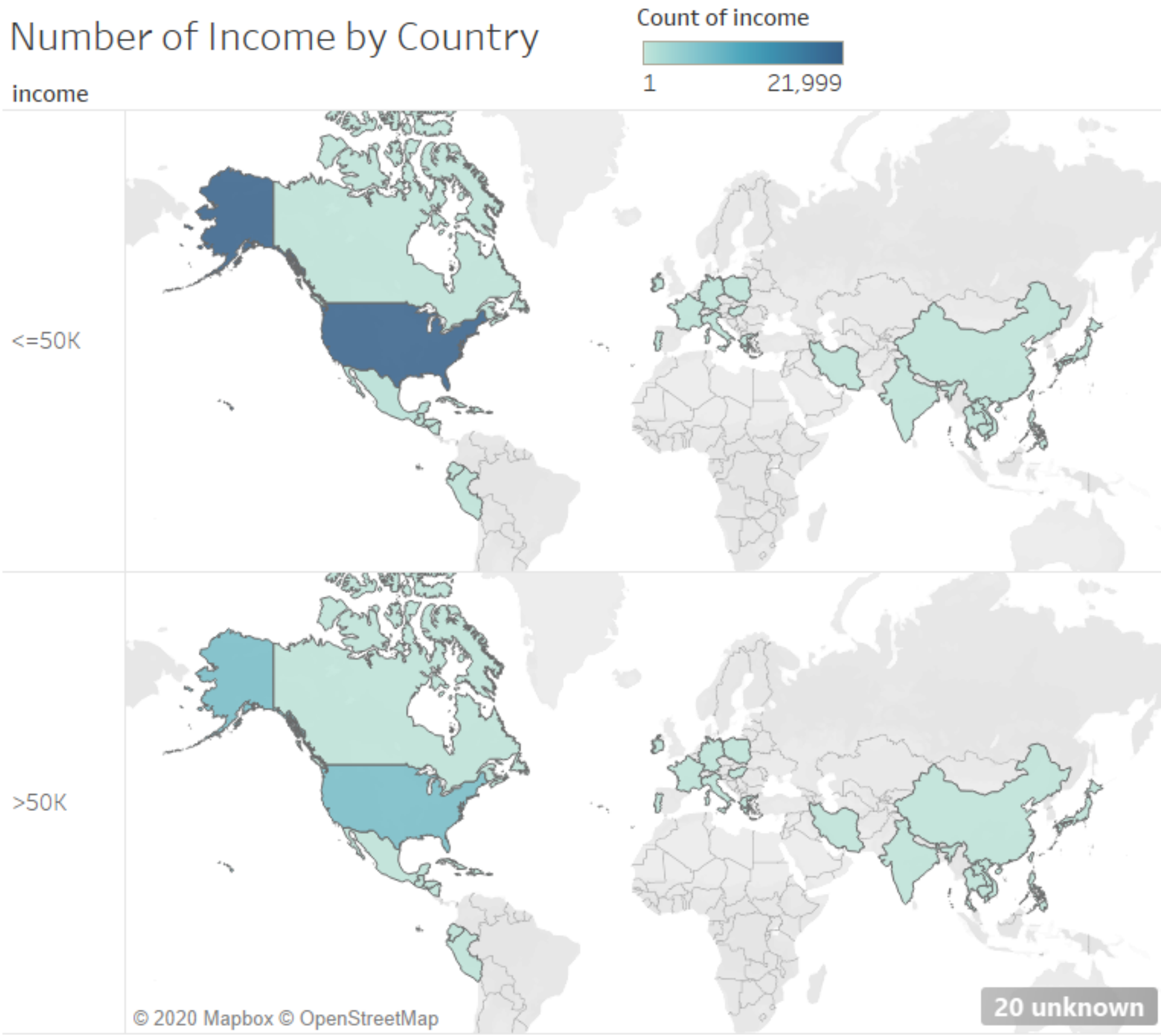
Number of Income by relationship



Number of Income by race




Number of Income by education level
income / education
<=50K
>50K
income 计数
8K
6K
4K
2K
0K
Some-college
10th
9th
Prof-school
Masters
Assoc-voc
Assoc-acdm
11th
12th
education
1st-4th
5th-6th
7th-8th
9th
10th
11th
12th
Assoc-acdm
Assoc-voc
Bachelors
Doctorate
HS-grad
Masters
Preschool
Prof-school
Some-college


Number of Income by Country
Count of income
1
21,999
income
<=50K
>50K
© 2020 Mapbox © OpenStreetMap
20 unknown

Python Notebook:

Using the python codes to handle the project, including data clean, EDA, feature selection, model building, hyper tuning, and data testing in the final model.

Below are our coding and graph explaining:

1. Checking the original data information with the data type, missing values, unique values

	Dtype	Nunique	MissingValues	Count	ZeroValues	?Values
id	int64	32561	0	32561	0	0
age	int64	73	0	32561	0	0
workclass	object	9	0	32561	0	0
fnlwgt	int64	21648	0	32561	0	0
education	object	16	0	32561	0	0
education_num	int64	16	0	32561	0	0
marital_status	object	7	0	32561	0	0
occupation	object	15	0	32561	0	0
relationship	object	6	0	32561	0	0
race	object	5	0	32561	0	0
sex	object	2	0	32561	0	0
capital_gain	int64	119	0	32561	29849	0

We can see that our dataset is clean, there are no missing values, but we do have some zero values. we will deal with it later.

2. Deal with the '?' value to null values.

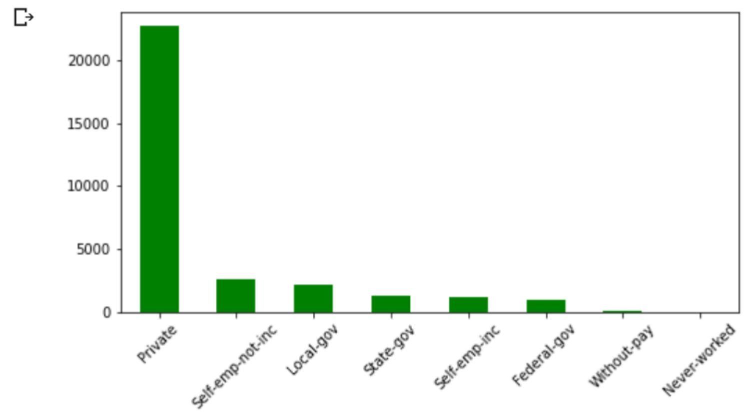
```
[53] # there is an extra space before each value of categorical column so correct it.
     for col in categorical_feature:
         df[col] = df[col].str.strip()
```

```
[54] # enocde '?' to nan value
     # df['workclass'].replace('?', np.NaN)
     df['workclass'] = np.where(df['workclass']=='?',np.NaN,df['workclass'])
```

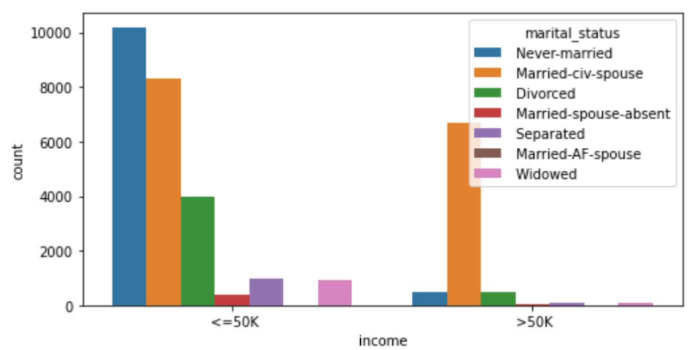
```
0                State-gov
1         Self-emp-not-inc
2                  Private
3                  Private
4                  Private
               ...
32556              Private
32557              Private
32558              Private
32559              Private
32560         Self-emp-inc
Name: workclass, Length: 32561, dtype: object
```

3. Check the unique values and value counts for each feature and draw plots

```
plt.figure(figsize=(8,4))
df['workclass'].value_counts().plot(kind='bar', color = 'green')
plt.xticks(rotation=45)
plt.show()
```

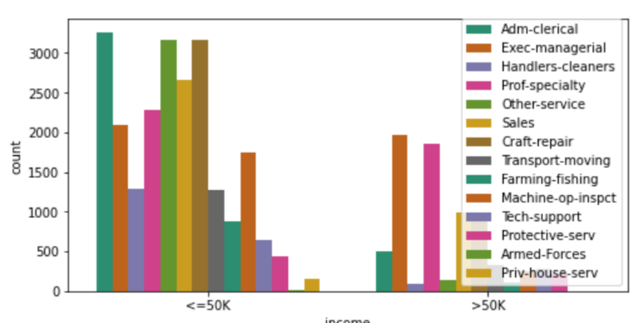



```
plt.figure(figsize=(8,4))
sns.countplot(x='income',hue='marital_status',data=df)
plt.show()
```

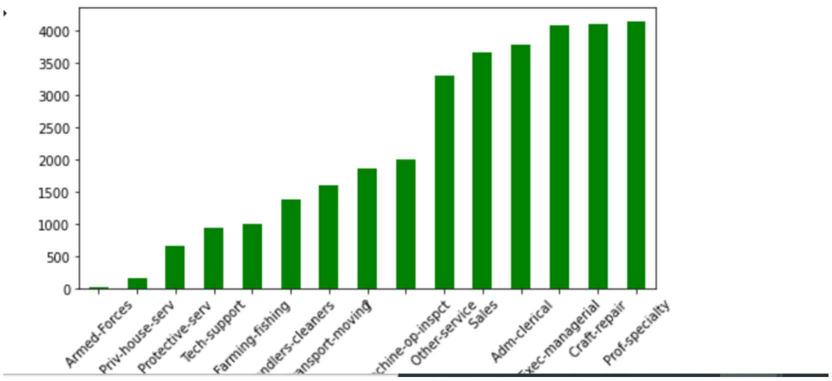



```
# same problem in occupation is there
df['occupation'] = np.where(df['occupation']=='?',np.NaN, df['occupation'])
```

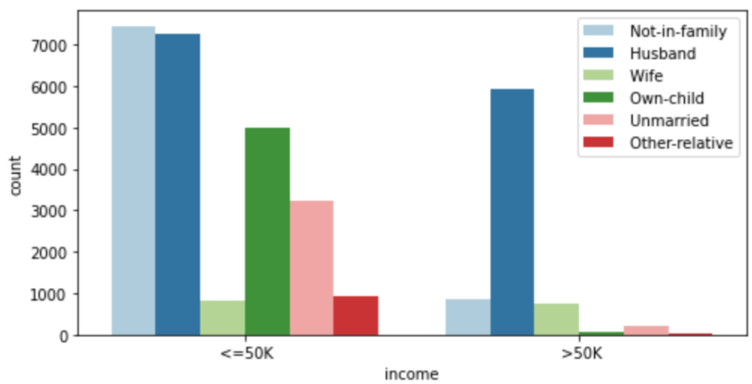
```
plt.figure(figsize=(8,4))
sns.countplot(x='income',hue='occupation',data=df,palette='Dark2')
plt.legend(loc='best')
plt.show()
```



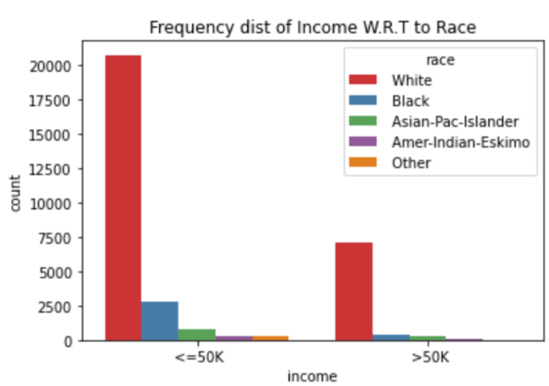
```python
plt.figure(figsize=(8,4))
df['occupation'].value_counts().sort_values().plot(kind='bar', color = 'green')
plt.xticks(rotation=45)
plt.show()
```




```python
plt.figure(figsize=(8,4))
sns.countplot(x='income',hue='relationship',data=df,palette='Paired')
plt.legend(loc='best')
plt.show()
```

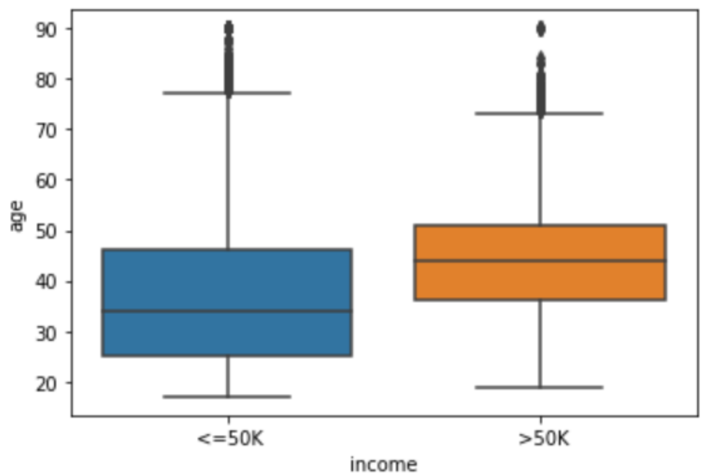



```python
# let's see income with respect to sex
plt.figure(figsize=(6,4))
sns.countplot(x='income',hue='race',data=df,palette='Set1')
plt.title("Frequency dist of Income W.R.T to Race")
plt.show()
```

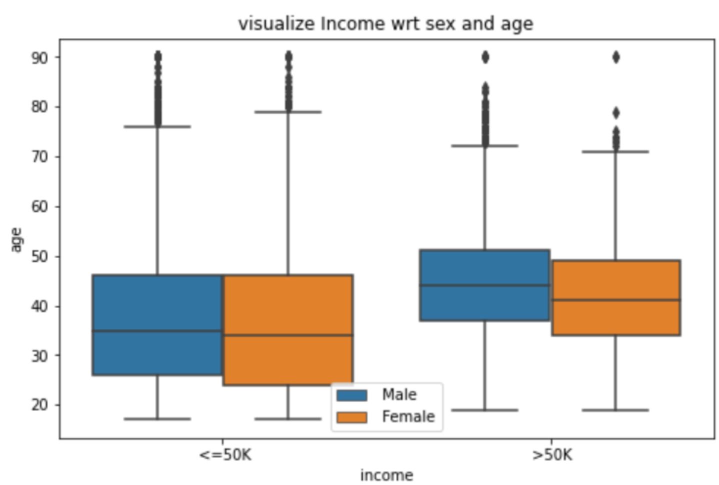


For numerical features plots:

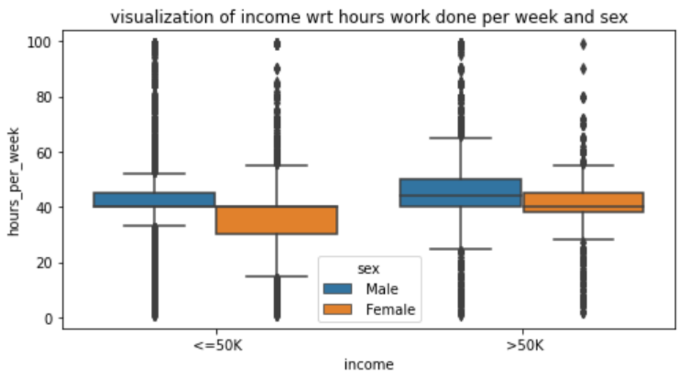
```python
# Income wrt age
plt.figure(figsize=(6,4))
sns.boxplot(x='income',y='age',data=df)
plt.show()
```



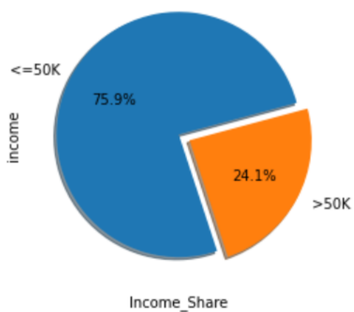

```python
# Income wrt age
plt.figure(figsize=(8,5))
ax = sns.boxplot(x='income',y='age',hue='sex',data=df)
ax.set_title("visualize Income wrt sex and age")
ax.legend(loc='best')
plt.show()
```



```python
plt.figure(figsize=(8,4))
ax = sns.boxplot(x='income',y='hours_per_week',hue='sex',data=df)
ax.set_title("visualization of income wrt hours work done per week and sex")
plt.show()
```




```python
plt.figure(figsize=(4,4))
df['income'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',shadow=True,startangle=15)
plt.xlabel("Income_Share")
plt.show()
```



4. Impute Missing Values: mode

```python
df[categorical_feature].isnull().sum()
```

```
workclass         1836
education            0
marital_status       0
occupation        1843
relationship         0
race                 0
sex                  0
native_country     583
income               0
dtype: int64
```

```
[63] # get the mmost frequency value in workclass

     df['workclass'].mode()[0]

     'Private'
```

So for the categorical data, in this case we can use Frequent Category Imputation and Random Sampling Imputation Both. both will work fine.

```
# use the most frequent value to impute the mising values

df['workclass'].fillna(df['workclass'].mode()[0], inplace=True)
df['occupation'].fillna(df['occupation'].mode()[0], inplace=True)
df['native_country'].fillna(df['native_country'].mode()[0], inplace=True)
```

5. Categorical features Encoding

```
[77] # map the label with mean values  in each column after group by
     # using the index after the sort values to map the labels
     for col in categorical:
         labels = df.groupby(col)['income'].mean().sort_values().index
         mapping_dict = {k: i for i, k in enumerate(labels, 0)}
         # apply encoding to our data
         df[col] = df[col].map(mapping_dict)
```

```
convert all the categorical data to numbers

df.head()
```

.d	age	workclass	fnlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hours_per_week	nat
1	39	3.0	77516	12	13	0	6.0	3	3	1	2174	0	40	
2	50	4.0	83311	12	13	6	13.0	4	3	1	0	0	13	
3	38	2.0	215646	8	9	4	2.0	3	3	1	0	0	40	
4	53	2.0	234721	3	7	6	2.0	4	2	1	0	0	40	

6. Feature Selection and feature scaling

Feature Scalling¶

we will use MinMaxScaler to handle the inbalance dataset

```
[105] # use the minmax to scale the dataset

      from sklearn.preprocessing import MinMaxScaler

      minmax = MinMaxScaler()

      X = minmax.fit_transform(x)
```

```
extra_tree = ExtraTreesClassifier(n_estimators=5, criterion='entropy', max_features=3)

extra_tree.fit(X,y)
```

```
ExtraTreesClassifier(bootstrap=False, ccp_alpha=0.0, class_weight=None,
                     criterion='entropy', max_depth=None, max_features=3,
                     max_leaf_nodes=None, max_samples=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=5, n_jobs=None,
                     oob_score=False, random_state=None, verbose=0,
```

7. Split data to train and test, and define a function to evaluate models

```
[118] X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=0)
```

```
# Create a function for Model Evaluation Using Cross-Validation
def cross_val_score_multilabel (model, X_train, y_train):
    accuracy_scores = cross_val_score(model, X_train, y_train, cv=5, scoring='accuracy')
    recall = cross_val_score(model, X_train, y_train, cv=5, scoring='recall')
    f1 = cross_val_score(model, X_train, y_train, cv=5, scoring='f1')
    precision = cross_val_score(model, X_train, y_train, cv=5, scoring='precision')
    roc_auc = cross_val_score(model, X_train, y_train, cv=5, scoring='roc_auc')
    print('Model Mean Accuracy (training set):{} '.format(np.mean(accuracy_scores)))
    print(' Recall (training set):{} '.format(np.mean(recall)))
    print(' F1 score (training set):{} '.format(np.mean(f1)))
    print(' precision (training set):{} '.format(np.mean(precision)))
    print(' roc_auc score (training set):{} '.format(np.mean(roc_auc)))
```

8. Model building and choose the best model
   For this final project, we would build logistic regression, KNN, SVM, decision tree, and random forest models and compare their results.

1) Compare all the Models evaluations:

Metric	Logistic Regression model	KNN model	SVM model	Decision Tree model	Random Forest model
The Accuracy of the model	0.8454	0.8247	0.8495	0.8144	0.9164
Recall	0.5672	0.5662	0.5395	0.6327	0.6305
F1 Score	0.6377	0.6077	0.6323	0.6205	0.6897
precision	0.7285	0.6560	0.7638	0.6087	0.7616
roc_aucroc_auc	0.8988	0.8444	0.892	0.7522	0.9128

5 Random Forest Model

```
rf = RandomForestClassifier(random_state=0)
rf.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

Evaluate model scores

```
# evaluate Random Forest model

cross_val_score_multilabel (rf, X_train, y_train)
```

```
Model Mean Accuracy (training set):0.863943212121056
 Recall (training set):0.6304848038430745
 F1 score (training set):0.6897094922111011
 precision (training set):0.7615822379743515
 roc_auc score (training set):0.9127662371731635
```

9. Hyperparameter tuning for Random Forest with RandomizedSearchCV and get the best evaluating scores

```
# tuned model:
rf02 = RandomForestClassifier(
                       n_estimators= 1400, min_samples_split= 2, min_samples_leaf= 4, max_features= 'sqrt', max_depth= 40, bootstrap= True)

rf02.fit(X_train,y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=40, max_features='sqrt',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=4, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=1400,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
# evaluate the tuned random forest model:
cross_val_score_multilabel (rf02, X_train, y_train)
```

```
Model Mean Accuracy (training set):0.8681661409143852
 Recall (training set):0.6250413450760608
 F1 score (training set):0.6938273343671518
 precision (training set):0.781457407940578
 roc_auc score (training set):0.9188595994019547
```

10. Randomly choose a record(id = 18) to test our model

3) Use the explainer to explain predictions

We chose instance 18 from the testing set as an example, and its attrition value is 0. Let's see what and how our model predicts this employee attrition.

```
# example
i=18

df_new.loc[[i]]
```

nlwgt	education	education_num	marital_status	occupation	relationship	race	sex	capital_gain	capital_loss	hours_per_week	native_country	income
28887	3	7	6	9.0	4	3	1	0	0	50	23.0	0

```
[155] # attrition actual value
y.loc[[i]]
```

```
18    0
Name: income, dtype: uint8
```

11. Our model result is the same as the original dataset result. Our best model works well with a 0.868 accuracy score