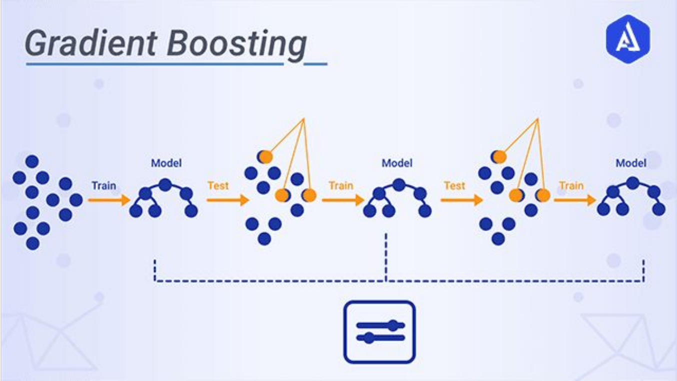
Income Level Prediction using Gradient Boosting Classifier

By Noah Vining

Objective

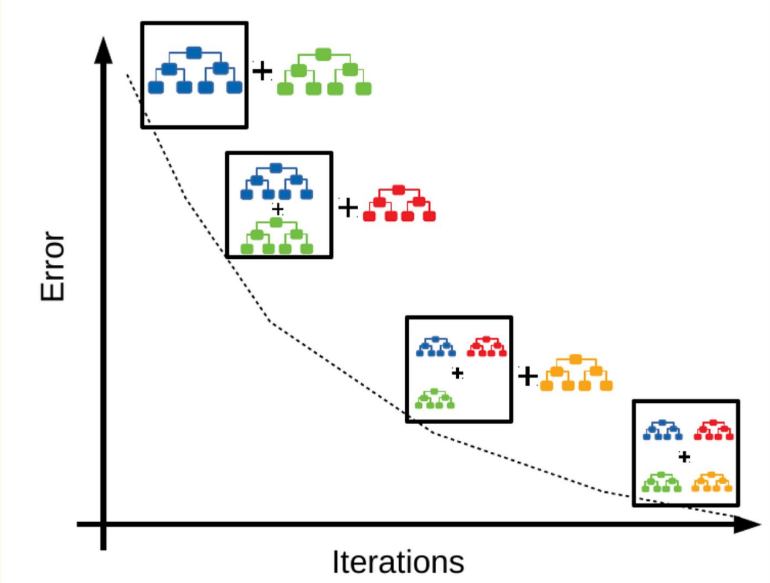
- Predict whether an individual's yearly income is above or below $50,000
- Using Census Data
- Using Gradient Boosting Classifier



	CountyId	State	County	TotalPop	Men	Women	Hispanic	White	Black	Native	Asian	Pacific	VotingAge(	Income	IncomeErr	IncomePe	IncomePe	Poverty	ChildPover	Profession	Service	Office	Constructi
2	1001	Alabama	Autauga Co	55036	26899	28137	2.7	75.4	18.9	0.3	0.9	0	41016	55317	2838	27824	2024	13.7	20.1	35.3	18	23.2	8.1
3	1003	Alabama	Baldwin Co	203360	99527	103833	4.4	83.1	9.5	0.8	0.7	0	155376	52562	1348	29364	735	11.8	16.1	35.7	18.2	25.6	9.7
4	1005	Alabama	Barbour Co	26201	13976	12225	4.2	45.7	47.8	0.2	0.6	0	20269	33368	2551	17561	798	27.2	44.9	25	16.8	22.6	11.5
5	1007	Alabama	Bibb Count	22580	12251	10329	2.4	74.6	22	0.4	0	0	17662	43404	3431	20911	1889	15.2	26.6	24.4	17.6	19.7	15.9
6	1009	Alabama	Blount Cou	57667	28490	29177	9	87.4	1.5	0.3	0.1	0	42513	47412	2630	22021	850	15.6	25.4	28.5	12.9	23.3	15.8
7	1011	Alabama	Bullock Co	10478	5616	4862	0.3	21.6	75.6	1	0.7	0	8212	29655	5376	20856	2355	28.5	50.4	19.7	17.1	18.6	14
8	1013	Alabama	Butler Cou	20126	9416	10710	0.3	52.2	44.7	0.1	1.1	0	15459	36326	2701	19004	943	24.4	34.8	26.9	17.3	18.5	11.6
9	1015	Alabama	Calhoun Co	115527	55593	59934	3.6	72.7	20.4	0.2	1	0	88383	43686	1491	23638	793	18.6	26.6	29	17.5	23.7	10.4
10	1017	Alabama	Chambers	33895	16320	17575	2.2	56.2	39.3	0.3	1	0	26259	37342	2011	22002	1205	18.8	29.1	24.3	13.5	23	11.6
11	1019	Alabama	Cherokee (	25855	12862	12993	1.6	91.8	5	0.5	0.1	0	20620	40041	2316	23010	1354	16.1	20	28.8	14.8	18.1	11.9
12	1021	Alabama	Chilton Co	43805	21554	22251	7.7	80.4	9.5	0.4	0.4	0	31776	43501	2877	23368	1925	19.4	27.8	25.3	14.5	23.7	15.5
13	1023	Alabama	Choctaw C	13188	6277	6911	0.5	56.3	42.1	0	0.1	0	10454	32122	2797	20994	1307	22.3	32.8	23.6	15.4	22	17.1
14	1025	Alabama	Clarke Cou	24625	11649	12976	0.2	53	45.7	0.1	0.5	0.1	19087	33827	2336	20765	1203	25.3	30.7	21.6	14.3	24.8	13.7
15	1027	Alabama	Clay Count	13407	6471	6936	3.1	80.2	14.7	0.9	0	0	10332	37287	6507	21330	1553	19.1	23.5	22.2	14.6	18.4	12.9
16	1029	Alabama	Cleburne (	14939	7367	7572	2.4	92.7	2.8	0.3	0.5	0	11329	37396	5390	20873	1251	19.1	31	25.7	11.4	23.2	18
17	1031	Alabama	Coffee Cou	51073	25225	25848	6.7	71	17.1	1	1.3	0	38026	49821	1945	26216	950	16.1	26.3	31.6	17.3	21.4	13.1
18	1033	Alabama	Colbert Co	54435	26143	28292	2.4	78.8	15.9	0.7	0.5	0	42195	45477	1959	23675	816	16.8	24.1	27.1	15.4	26	11.2
19	1035	Alabama	Conecuh (	12649	6040	6609	1.8	50.3	46.3	0.2	0	0	9812	30434	3446	16337	934	26.4	39.3	15.9	19.7	24.3	12.8
20	1037	Alabama	Coosa Cou	10955	5474	5481	0.1	65.3	33.2	0.1	0	0	8989	34792	3587	20342	1728	14.4	17.9	17.6	23.2	23.7	14.5
21	1039	Alabama	Covington (	37519	18133	19386	1.6	83.5	13	0.3	0.4	0	29164	39467	2392	22431	830	17.6	25	29.2	14.3	22.2	15.8
22	1041	Alabama	Crenshaw	13866	6786	7080	1.8	71.2	23.1	0.8	1.4	0	10551	38937	3177	21580	1212	17.6	22.5	26.4	13.1	24.1	13.2
23	1043	Alabama	Cullman Co	81703	40223	41480	4.2	92.3	1.1	0.4	0.6	0	62012	40997	1715	21857	729	16.4	19.6	29.3	14.6	23.4	14
24	1045	Alabama	Dale Count	49393	24524	24869	6.1	69.5	19.2	0.5	0.9	0	37173	44711	2010	23194	868	19.6	27.2	28.2	15.3	26.6	13.7
25	1047	Alabama	Dallas Cou	40755	18809	21946	1	27.7	70.2	0	0.4	0	30303	30065	1775	18248	759	31.9	49.5	26.7	18.2	20.9	8.8
26	1049	Alabama	DeKalb Co	71194	35328	35866	14.3	80.8	1.4	1.2	0.3	0.1	49579	38842	2154	20020	899	21.5	31	23.7	15.2	20.8	12.4

What is a Gradient Boosting Classifier?

- Ensemble Learning
- Combines weak learners into strong learners.
- Is made up of decision trees
- Trains models sequentially, with each new model learning from the previous model.
- Each new model is trained to minimize loss function of previous model.
- Uses gradient to do minimize loss function.



Preprocessing:

- Import necessary libraries
    - Pandas
    - Numpy
    - Sklearn
    - Matplotlib
- Read in the data
    - Encoded in latin-1

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

from sklearn import metrics
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

```
census_data = pd.read_csv('Census.csv', encoding = 'latin-1')
census_data.head()
```

Preprocessing:

- Fill in missing values
  - data.fillna()
- Create X and Y Datasets
- Change Y Dataset
  - Binary representation of whether data is above 50,000 or below
- Split the datasets
  - train_test_split

```python
# Fill in missing variables with 0
census_data.fillna(0, inplace=True)

# Create X and Y datasets
census_X = census_data.drop(['Income', 'IncomeErr', 'State', 'County', 'Pacific', 'VotingAgeCitizen', 'Carpool'], axis=1)
census_Y = census_data['Income']

# Change Y dataset to be binary values based on being above or below 50000
census_Y = census_Y.apply(lambda x: 1 if x >= 50000 else 0)

# Split the data
train_X, test_X, train_y, test_y = train_test_split(census_X, census_Y, test_size = 0.25, random_state = 1)
```

Model Training

- Hyperparameters
  - N_estimators: Number of Decision Trees
  - Learning_rate: How much the model learns each iteration
  - Max_Depth: The maximum depth of each decision tree.
- Hyperparameter Tuning
  - Randomized Search
  - Cross-Validation

```
param_dist = {
    'n_estimators': np.arange(50, 251, 50),
    'learning_rate': np.linspace(0.01, 0.2, 10),
    'max_depth': np.arange(3, 8),
}

gradientBoost = GradientBoostingClassifier()

random_search = RandomizedSearchCV(estimator=gradientBoost, param_distributions=param_dist, n_iter=10,
                                   cv=5, scoring='accuracy', random_state=42, n_jobs=-1)

random_search.fit(train_X, train_y)

best_params_random = random_search.best_params_

print("Best Parameters (Randomized Search)", best_params_random)
```

Model Training

- Uses Hyperparameters found previously
- Fits model to train_X and train_Y
- Makes prediction using test_X

```
gradientBoost = GradientBoostingClassifier(n_estimators=100, max_depth=4, learning_rate= 0.15777777777777778)

gradientBoost.fit(train_X, train_y)

y_pred = gradientBoost.predict(test_X)

accuracy = accuracy_score(test_y, y_pred)

print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

Evaluation Metrics

- Accuracy
- Confusion Matrix
- Classification Report
  - Accuracy
  - Precision
  - Recall
  - F1 Score

```
accuracy = accuracy_score(test_y, y_pred)

print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
confusion_matrix = metrics.confusion_matrix(test_y, y_pred)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [0, 1])

cm_display.plot()

print(classification_report(test_y, y_pred))
```

Feature Importance

- Feature Importance Function
- Bar Plot
  - Census Data Columns as X
  - Feature Importance as Y
  - Plotted in descending order

```python
# Feature Importance:
x = census_X.columns
y = gradientBoost.feature_importances_ * 100

# Create a DataFrame to store feature importance
feature_importance_df = pd.DataFrame({'Feature': x, 'Importance': y})

# Sort the DataFrame by importance in descending order
feature_importance_df = feature_importance_df.sort_values('Importance', ascending=False)

fig, ax = plt.subplots(figsize =(18, 10))
plt.barh(feature_importance_df['Feature'], feature_importance_df['Importance'], color='blue')
plt.xlabel("Variable Importance")
plt.gca().invert_yaxis()
plt.show()
```