jupyter **Machine learning Project** Last Checkpoint: 04/11/2023 (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

Run | Code

```
# Initialize a scaler, then apply it to the features
scaler = MinMaxScaler() # default=(0, 1)
numerical = ['age', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week']
features_raw[numerical] = scaler.fit_transform(data[numerical])


# Show an example of a record with scaling applied
display(features_raw.head(n = 5))
```

	age	workclass	education_level	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
0	0.301370	State-gov	Bachelors	0.800000	Never-married	Adm-clerical	Not-in-family	White	Male	0.02174	0.0	0.397959	United-States
1	0.452055	Self-emp-not-inc	Bachelors	0.800000	Married-civ-spouse	Exec-managerial	Husband	White	Male	0.00000	0.0	0.122449	United-States
2	0.287671	Private	HS-grad	0.533333	Divorced	Handlers-cleaners	Not-in-family	White	Male	0.00000	0.0	0.397959	United-States
3	0.493151	Private	11th	0.400000	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0.00000	0.0	0.397959	United-States
4	0.150685	Private	Bachelors	0.800000	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0.00000	0.0	0.397959	Cuba

In [30]:

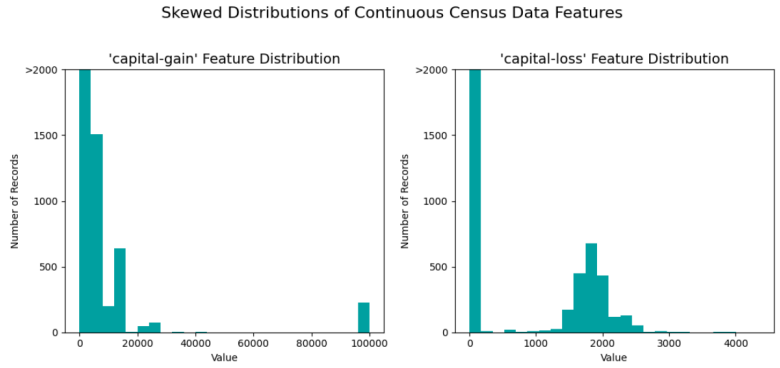
```
# TODO: One-hot encode the 'features_log_minmax_transform' data using pandas.get_dummies()
features_final = pd.get_dummies(features_raw)

# TODO: Encode the 'income_raw' data to numerical values
income = income_raw.apply(lambda x: 0 if x == '<=50K' else 1)

# Print the number of features after one-hot encoding
encoded = list(features_final.columns)
print("{} total features after one-hot encoding.".format(len(encoded)))
```


jupyter **Machine learning Project** Last Checkpoint: 04/11/2023 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3 (ipykernel)

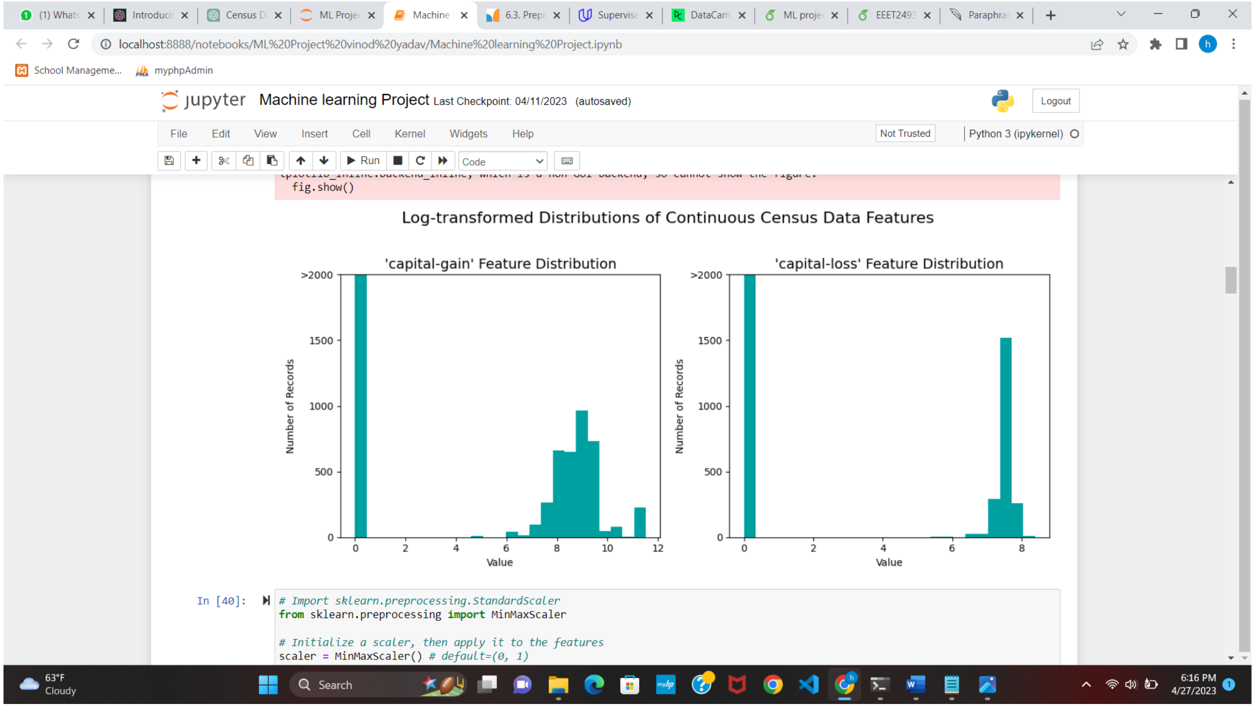
Run | Code

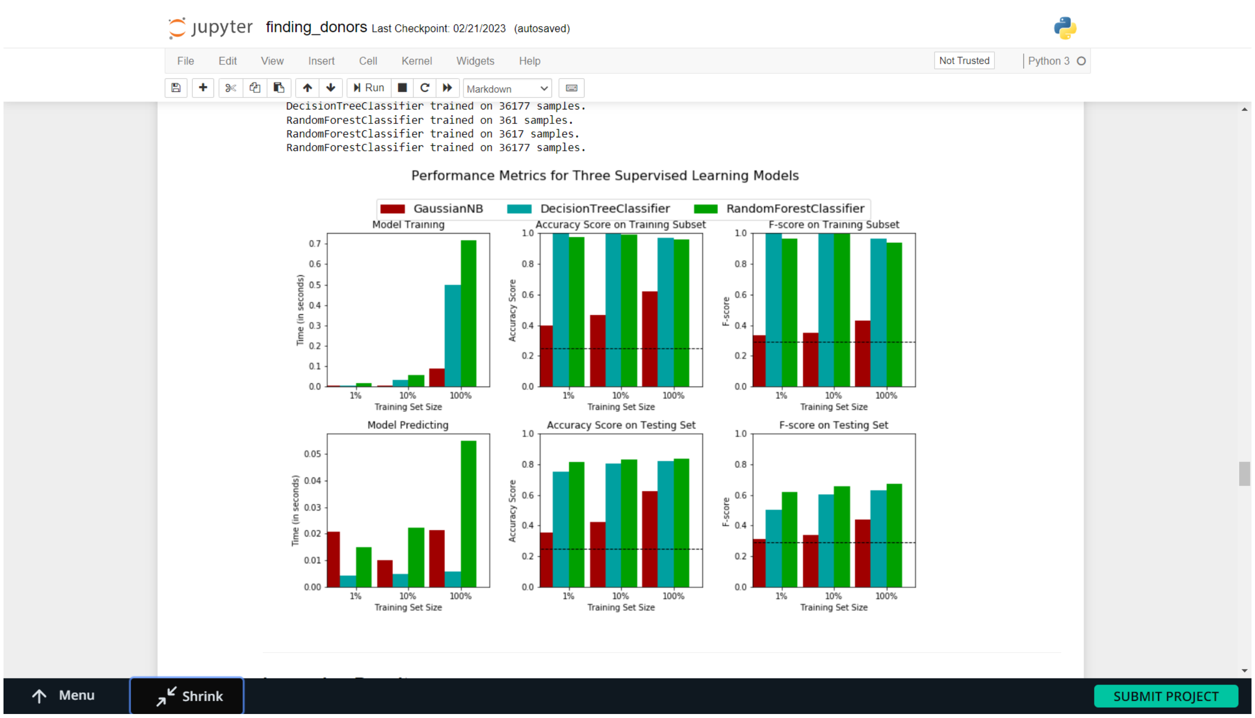
```
C:\Users\harsh\Jupyter notebook\ML Project vinod yadav\visuals.py:59: UserWarning: Matplotlib is currently using module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```



In [28]:

```
# Log-transform the skewed features
skewed = ['capital-gain', 'capital-loss']
features_raw[skewed] = data[skewed].apply(lambda x: np.log(x + 1))
```


localhost:8888/notebooks/ML%20Project%20vinod%20yadav/Machine%20learning%20Project.ipynb
Jupyter Machine learning Project Last Checkpoint: 04/11/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
fig.show()
Log-transformed Distributions of Continuous Census Data Features
'capital-gain' Feature Distribution
'capital-loss' Feature Distribution
Number of Records
Value
In [40]:
# Import sklearn.preprocessing.StandardScaler
from sklearn.preprocessing import MinMaxScaler
# Initialize a scaler, then apply it to the features
scaler = MinMaxScaler() # default=(0, 1)
63°F
Cloudy
Search
6:16 PM
4/27/2023

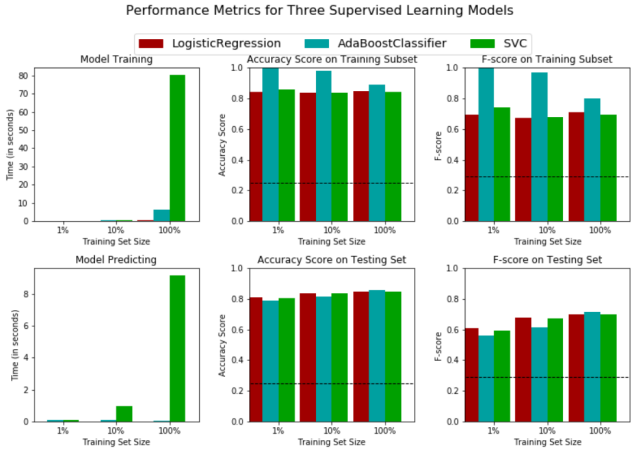

Jupyter finding_donors Last Checkpoint: 02/21/2023 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
DecisionTreeClassifier trained on 36177 samples.
RandomForestClassifier trained on 361 samples.
RandomForestClassifier trained on 3617 samples.
RandomForestClassifier trained on 36177 samples.
Performance Metrics for Three Supervised Learning Models
GaussianNB
DecisionTreeClassifier
RandomForestClassifier
Model Training
Accuracy Score on Training Subset
F-score on Training Subset
Model Predicting
Accuracy Score on Testing Set
F-score on Testing Set
Time (in seconds)
Accuracy Score
F-score
Training Set Size
Menu
Shrink
SUBMIT PROJECT

jupyter finding_donors Last Checkpoint: 02/21/2023 (autosaved)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

Run | Markdown

```
AdaBoostClassifier trained on 361 samples.
AdaBoostClassifier trained on 3617 samples.
AdaBoostClassifier trained on 36177 samples.
SVC trained on 361 samples.
SVC trained on 3617 samples.
SVC trained on 36177 samples.
```



Menu Shrink SUBMIT PROJECT

ML Project vinod yadav/ | Machine learning Project - | Introducing ChatGPT | New chat | Supervised Learning Projec | vinodyadav2413/Machine_ | localhost:8888/notebooks/ML%20Project%20vinod%20yadav/Machine%20learning%20Project.ipynb

School Manageme... myphpAdmin

jupyter Machine learning Project Last Checkpoint: 04/11/2023 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help | Trusted | Python 3 (ipykernel)

Run | Code

In [2]:
```
import numpy as np
import pandas as pd
from time import time
from IPython.display import display # Allows the use of display() for DataFrames


import visuals as vs


%matplotlib inline


data = pd.read_csv("census.csv")


display(data.head(n=1))
data['income'].value_counts()
```

	age	workclass	education_level	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	Bachelors	13.0	Never-married	Adm-clerical	Not-in-family	White	Male	2174.0	0.0	40.0	United-States	<=50K

Out[2]:
```
income
<=50K    34014
>50K     11208
Name: count, dtype: int64
```

In [3]:
```
# Total number of records

n_records =45222
```

62°F Cloudy | Search | 11:44 AM 5/5/2023

```
<=50K    34014
>50K     11208
Name: count, dtype: int64
```

In [3]:
```
# Total number of records

n_records =45222

# Number of records where individual's income is more than $50,000
n_greater_50k = 11208

#  Number of records where individual's income is at most $50,000
n_at_most_50k = 34014

# Percentage of individuals whose income is more than $50,000
greater_percent = 24.78

# Print the results
print("Total number of records: {}".format(n_records))
print("Individuals making more than $50,000: {}".format(n_greater_50k))
print("Individuals making at most $50,000: {}".format(n_at_most_50k))
print("Percentage of individuals making more than $50,000: {}%".format(greater_percent))
```

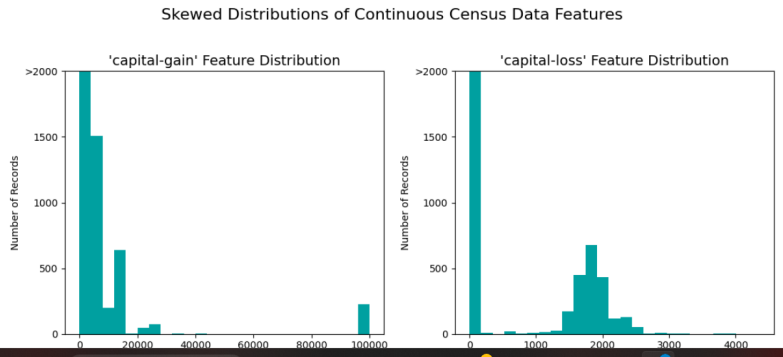
```
Total number of records: 45222
Individuals making more than $50,000: 11208
Individuals making at most $50,000: 34014
Percentage of individuals making more than $50,000: 24.78%
```

In [4]:
```
# Split the data into features and target label
income_raw = data['income']
features_raw = data.drop('income', axis = 1)

# Visualize skewed continuous features of original data
vs.distribution(data)
```


ML Project vinod yadav/ | Machine learning Project | Introducing ChatGPT | New chat | Supervised Learning Projec | vinodyadav2413/Machine_
localhost:8888/notebooks/ML%20Project%20vinod%20yadav/Machine%20learning%20Project.ipynb
School Manageme... myphpAdmin
jupyter Machine learning Project Last Checkpoint: 04/11/2023 (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)
Run Code

In [4]:
```
# Split the data into features and target label
income_raw = data['income']
features_raw = data.drop('income', axis = 1)

# Visualize skewed continuous features of original data
vs.distribution(data)
```

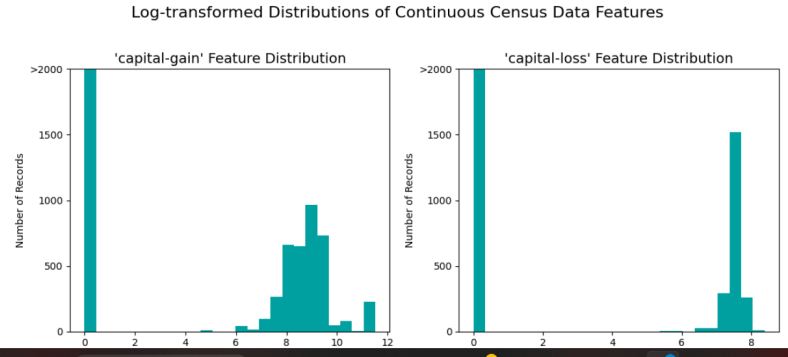
```
C:\Users\harsh\Jupyter notebook\ML Project vinod yadav\visuals.py:59: UserWarning: Matplotlib is currently using module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```



62°F Cloudy Search 11:45 AM 5/5/2023

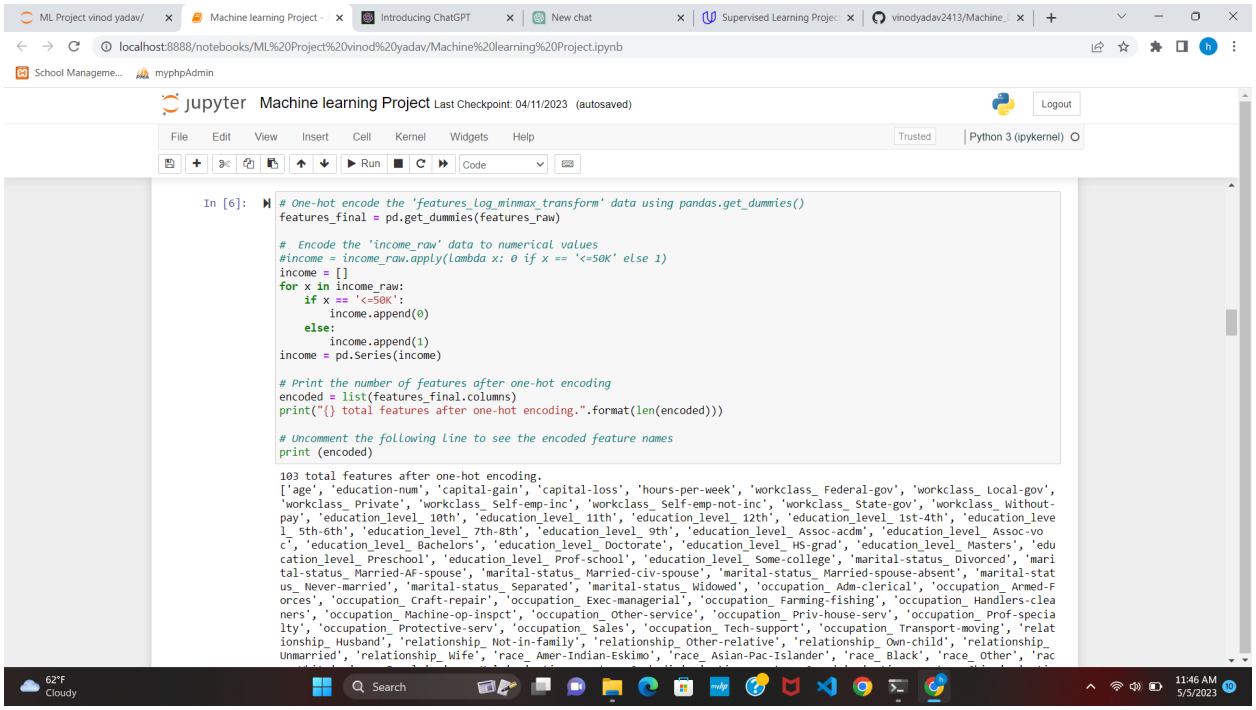
```
features_log_transformed = pd.DataFrame(data = features_raw)
features_log_transformed[skewed] = features_raw[skewed].apply(lambda x: np.log(x + 1))

# Visualize the new log distributions
vs.distribution(features_raw, transformed = True)
```

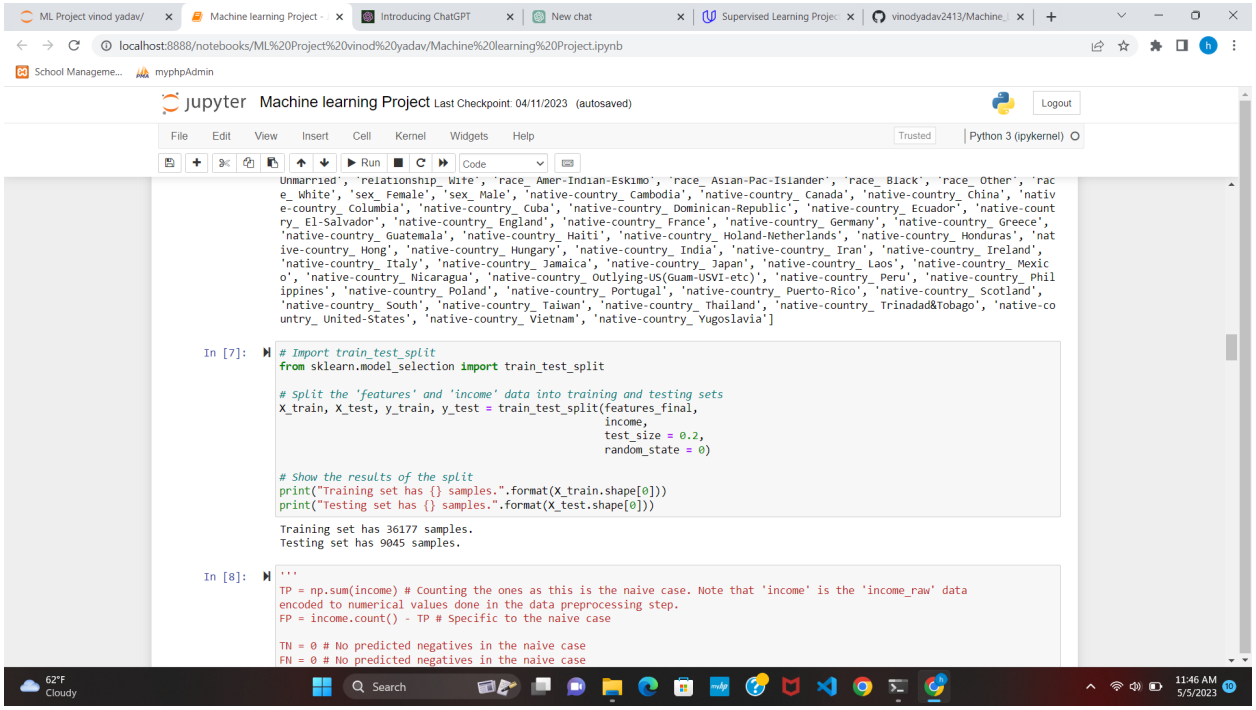
```
C:\Users\harsh\Jupyter notebook\ML Project vinod yadav\visuals.py:59: UserWarning: Matplotlib is currently using module://matplotlib_inline.backend_inline, which is a non-GUI backend, so cannot show the figure.
  fig.show()
```

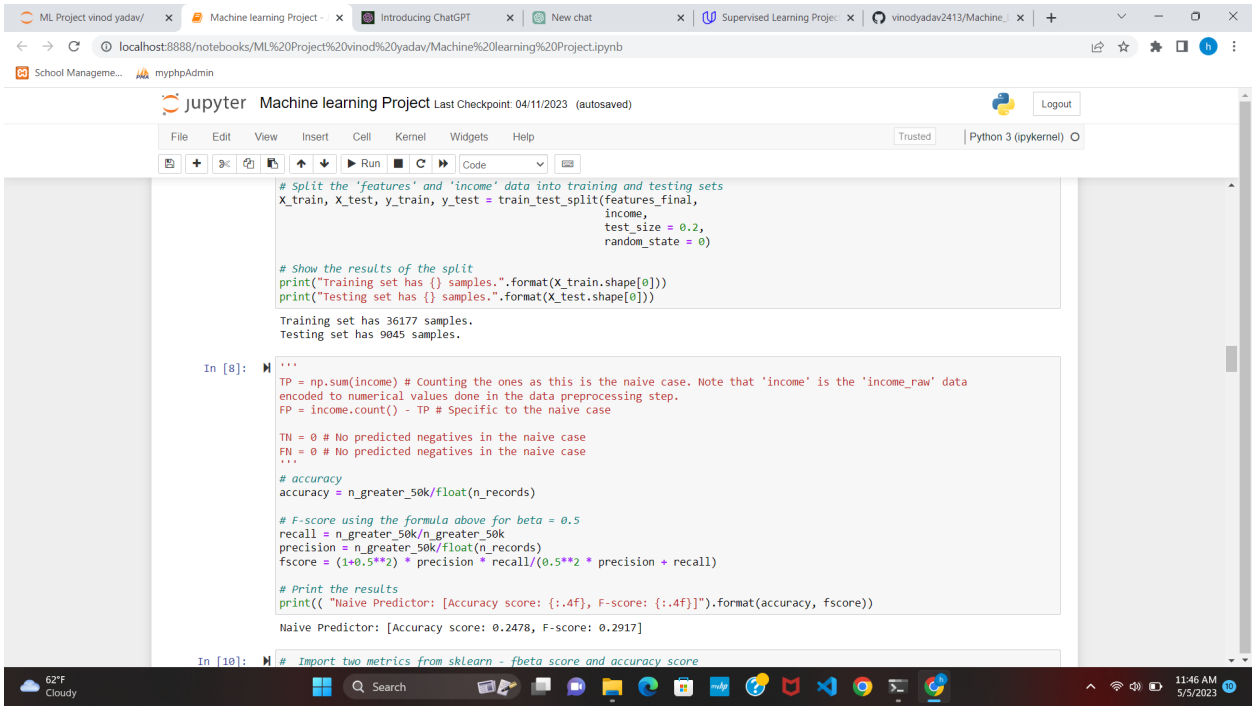



```
# Import sklearn.preprocessing.StandardScaler
from sklearn.preprocessing import MinMaxScaler

# Initialize a scaler, then apply it to the features
scaler = MinMaxScaler() # default=(0, 1)
numerical = ['age', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week']
features_raw[numerical] = scaler.fit_transform(data[numerical])


# Show an example of a record with scaling applied
display(features_raw.head(n = 5))
```

	age	workclass	education_level	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
0	0.301370	State-gov	Bachelors	0.800000	Never-married	Adm-clerical	Not-in-family	White	Male	0.02174	0.0	0.397959	United-States
1	0.452055	Self-emp-not-inc	Bachelors	0.800000	Married-civ-spouse	Exec-managerial	Husband	White	Male	0.00000	0.0	0.122449	United-States
2	0.287671	Private	HS-grad	0.533333	Divorced	Handlers-cleaners	Not-in-family	White	Male	0.00000	0.0	0.397959	United-States
3	0.493151	Private	11th	0.400000	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0.00000	0.0	0.397959	United-States
4	0.150685	Private	Bachelors	0.800000	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0.00000	0.0	0.397959	Cuba


ML Project vinod yadav/
Machine learning Project -
Introducing ChatGPT
New chat
Supervised Learning Projec
vinodyadav2413/Machine_
localhost:8888/notebooks/ML%20Project%20vinod%20yadav/Machine%20learning%20Project.ipynb
School Manageme...
myphpAdmin
jupyter Machine learning Project Last Checkpoint: 04/11/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [6]:
# One-hot encode the 'features_log_minmax_transform' data using pandas.get_dummies()
features_final = pd.get_dummies(features_raw)
# Encode the 'income_raw' data to numerical values
#income = income_raw.apply(lambda x: 0 if x == '<=50K' else 1)
income = []
for x in income_raw:
if x == '<=50K':
income.append(0)
else:
income.append(1)
income = pd.Series(income)
# Print the number of features after one-hot encoding
encoded = list(features_final.columns)
print("{} total features after one-hot encoding.".format(len(encoded)))
# Uncomment the following line to see the encoded feature names
print (encoded)
103 total features after one-hot encoding.
['age', 'education-num', 'capital-gain', 'capital-loss', 'hours-per-week', 'workclass_ Federal-gov', 'workclass_ Local-gov',
'workclass_ Private', 'workclass_ Self-emp-inc', 'workclass_ Self-emp-not-inc', 'workclass_ State-gov', 'workclass_ Without-
pay', 'education_level_ 10th', 'education_level_ 11th', 'education_level_ 12th', 'education_level_ 1st-4th', 'education_leve
l_ 5th-6th', 'education_level_ 7th-8th', 'education_level_ 9th', 'education_level_ Assoc-acdm', 'education_level_ Assoc-vo
c', 'education_level_ Bachelors', 'education_level_ Doctorate', 'education_level_ HS-grad', 'education_level_ Masters', 'edu
cation_level_ Preschool', 'education_level_ Prof-school', 'education_level_ Some-college', 'marital-status_ Divorced', 'mari
tal-status_ Married-AF-spouse', 'marital-status_ Married-civ-spouse', 'marital-status_ Married-spouse-absent', 'marital-stat
us_ Never-married', 'marital-status_ Separated', 'marital-status_ Widowed', 'occupation_ Adm-clerical', 'occupation_ Armed-F
orces', 'occupation_ Craft-repair', 'occupation_ Exec-managerial', 'occupation_ Farming-fishing', 'occupation_ Handlers-clea
ners', 'occupation_ Machine-op-inspct', 'occupation_ Other-service', 'occupation_ Priv-house-serv', 'occupation_ Prof-specia
lty', 'occupation_ Protective-serv', 'occupation_ Sales', 'occupation_ Tech-support', 'occupation_ Transport-moving', 'relat
ionship_ Husband', 'relationship_ Not-in-family', 'relationship_ Other-relative', 'relationship_ Own-child', 'relationship_
Unmarried', 'relationship_ Wife', 'race_ Amer-Indian-Eskimo', 'race_ Asian-Pac-Islander', 'race_ Black', 'race_ Other', 'rac
62°F Cloudy
Search
11:46 AM 5/5/2023


ML Project vinod yadav/
Machine learning Project -
Introducing ChatGPT
New chat
Supervised Learning Projec
vinodyadav2413/Machine_
localhost:8888/notebooks/ML%20Project%20vinod%20yadav/Machine%20learning%20Project.ipynb
School Manageme...
myphpAdmin
jupyter Machine learning Project Last Checkpoint: 04/11/2023 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Unmarried', 'relationship_ Wife', 'race_ Amer-Indian-Eskimo', 'race_ Asian-Pac-Islander', 'race_ Black', 'race_ Other', 'rac
e_ White', 'sex_ Female', 'sex_ Male', 'native-country_ Cambodia', 'native-country_ Canada', 'native-country_ China', 'nativ
e-country_ Columbia', 'native-country_ Cuba', 'native-country_ Dominican-Republic', 'native-country_ Ecuador', 'native-count
ry_ El-Salvador', 'native-country_ England', 'native-country_ France', 'native-country_ Germany', 'native-country_ Greece',
'native-country_ Guatemala', 'native-country_ Haiti', 'native-country_ Holand-Netherlands', 'native-country_ Honduras', 'nat
ive-country_ Hong', 'native-country_ Hungary', 'native-country_ India', 'native-country_ Iran', 'native-country_ Ireland',
'native-country_ Italy', 'native-country_ Jamaica', 'native-country_ Japan', 'native-country_ Laos', 'native-country_ Mexic
o', 'native-country_ Nicaragua', 'native-country_ Outlying-US(Guam-USVI-etc)', 'native-country_ Peru', 'native-country_ Phil
ippines', 'native-country_ Poland', 'native-country_ Portugal', 'native-country_ Puerto-Rico', 'native-country_ Scotland',
'native-country_ South', 'native-country_ Taiwan', 'native-country_ Thailand', 'native-country_ Trinadad&Tobago', 'native-co
untry_ United-States', 'native-country_ Vietnam', 'native-country_ Yugoslavia']
In [7]:
# Import train_test_split
from sklearn.model_selection import train_test_split
# Split the 'features' and 'income' data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features_final,
income,
test_size = 0.2,
random_state = 0)
# Show the results of the split
print("Training set has {} samples.".format(X_train.shape[0]))
print("Testing set has {} samples.".format(X_test.shape[0]))
Training set has 36177 samples.
Testing set has 9045 samples.
In [8]:
'''
TP = np.sum(income) # Counting the ones as this is the naive case. Note that 'income' is the 'income_raw' data
encoded to numerical values done in the data preprocessing step.
FP = income.count() - TP # Specific to the naive case
TN = 0 # No predicted negatives in the naive case
FN = 0 # No predicted negatives in the naive case
62°F Cloudy
Search
11:46 AM 5/5/2023



```
# Split the 'features' and 'income' data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features_final,
                                                    income,
                                                    test_size = 0.2,
                                                    random_state = 0)

# Show the results of the split
print("Training set has {} samples.".format(X_train.shape[0]))
print("Testing set has {} samples.".format(X_test.shape[0]))
```

```
Training set has 36177 samples.
Testing set has 9045 samples.
```

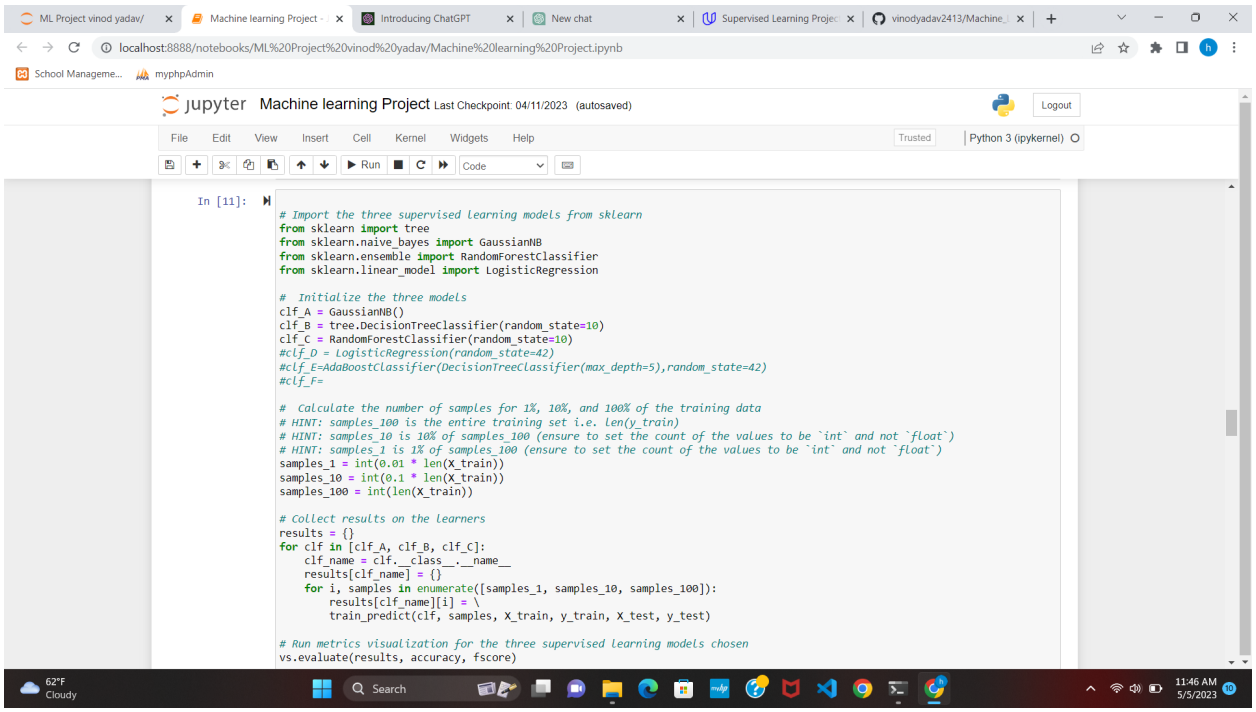
In [8]:

```
'''
TP = np.sum(income) # Counting the ones as this is the naive case. Note that 'income' is the 'income_raw' data
encoded to numerical values done in the data preprocessing step.
FP = income.count() - TP # Specific to the naive case

TN = 0 # No predicted negatives in the naive case
FN = 0 # No predicted negatives in the naive case
'''
# accuracy
accuracy = n_greater_50k/float(n_records)

# F-score using the formula above for beta = 0.5
recall = n_greater_50k/n_greater_50k
precision = n_greater_50k/float(n_records)
fscore = (1+0.5**2) * precision * recall/(0.5**2 * precision + recall)

# Print the results
print(( "Naive Predictor: [Accuracy score: {:.4f}, F-score: {:.4f}]").format(accuracy, fscore))
```

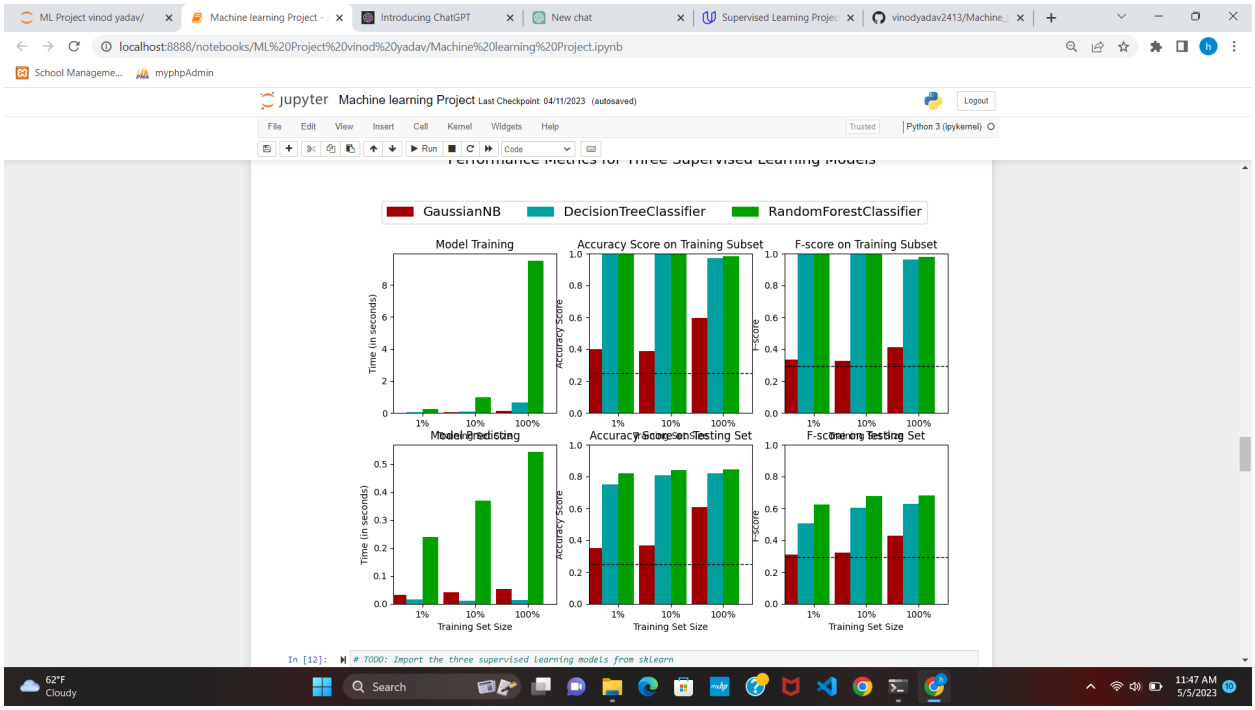
```
Naive Predictor: [Accuracy score: 0.2478, F-score: 0.2917]
```

In [10]:

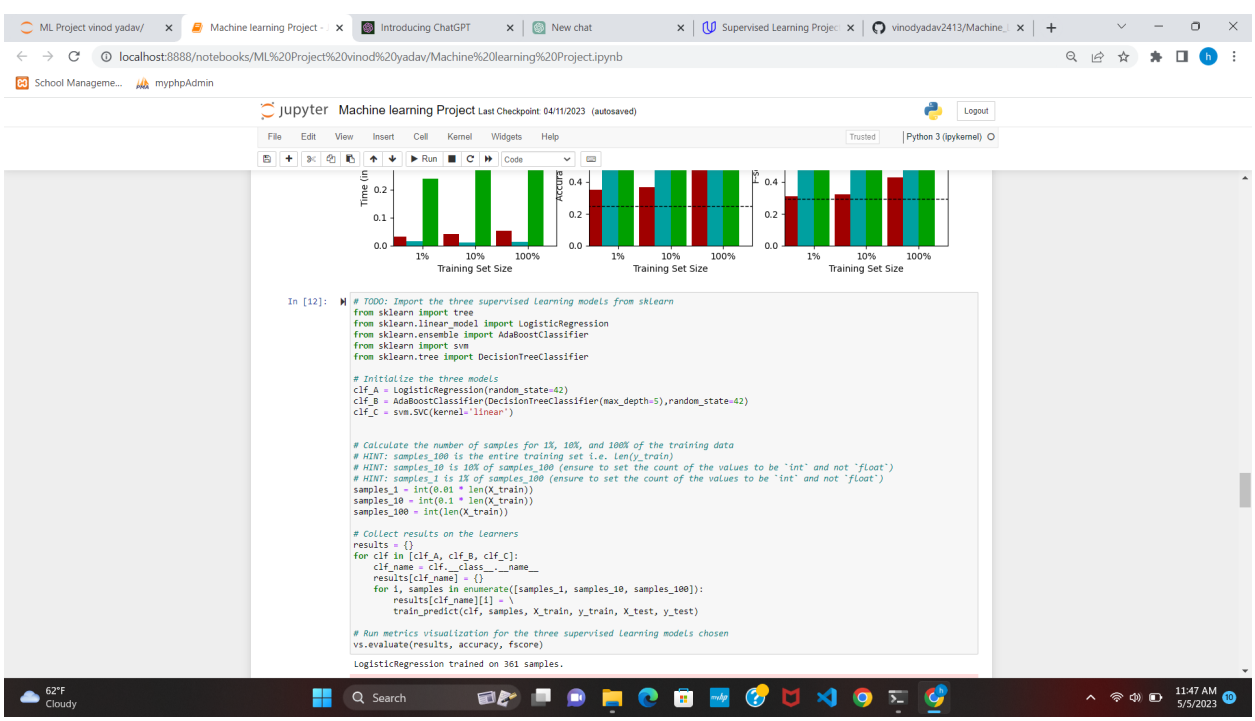
```
#  Import two metrics from sklearn - fbeta_score and accuracy_score
```



```
# Import the three supervised learning models from sklearn
from sklearn import tree
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

#  Initialize the three models
clf_A = GaussianNB()
clf_B = tree.DecisionTreeClassifier(random_state=10)
clf_C = RandomForestClassifier(random_state=10)
#clf_D = LogisticRegression(random_state=42)
#clf_E=AdaBoostClassifier(DecisionTreeClassifier(max_depth=5),random_state=42)
#clf_F=

#  Calculate the number of samples for 1%, 10%, and 100% of the training data
# HINT: samples_100 is the entire training set i.e. len(y_train)
# HINT: samples_10 is 10% of samples_100 (ensure to set the count of the values to be `int` and not `float`)
# HINT: samples_1 is 1% of samples_100 (ensure to set the count of the values to be `int` and not `float`)
samples_1 = int(0.01 * len(X_train))
samples_10 = int(0.1 * len(X_train))
samples_100 = int(len(X_train))

# Collect results on the learners
results = {}
for clf in [clf_A, clf_B, clf_C]:
    clf_name = clf.__class__.__name__
    results[clf_name] = {}
    for i, samples in enumerate([samples_1, samples_10, samples_100]):
        results[clf_name][i] = \
        train_predict(clf, samples, X_train, y_train, X_test, y_test)

# Run metrics visualization for the three supervised learning models chosen
vs.evaluate(results, accuracy, fscore)
```

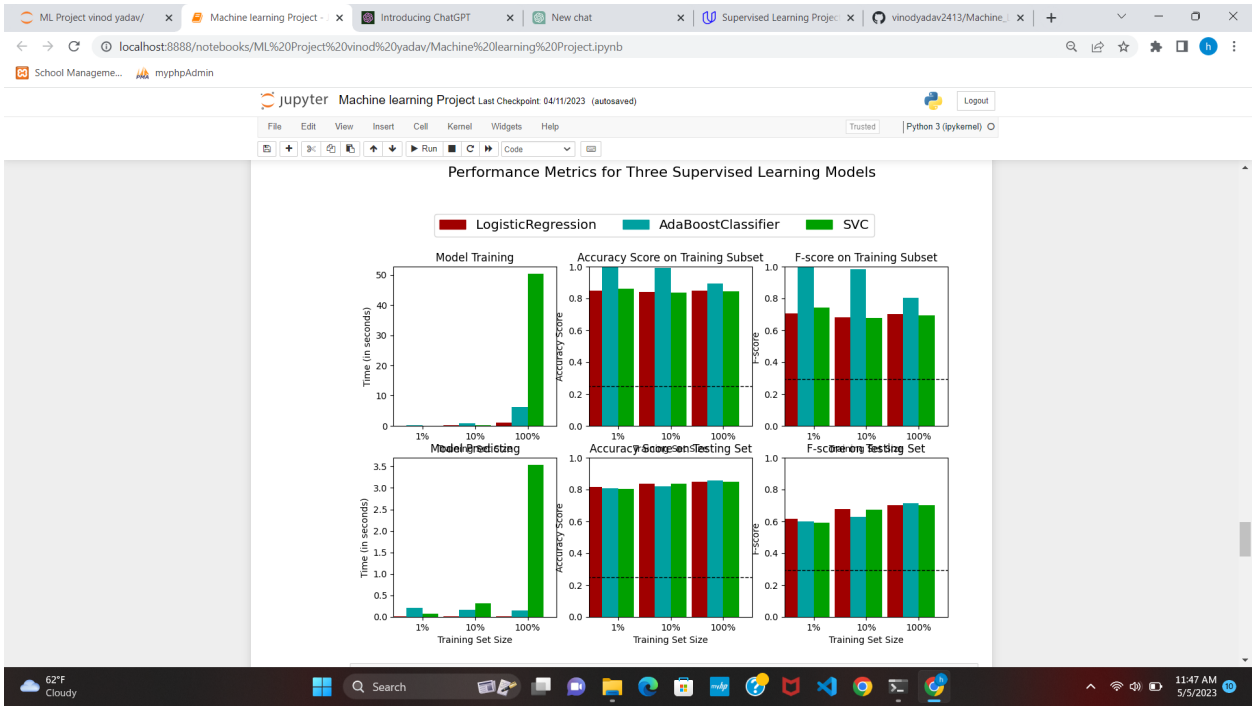


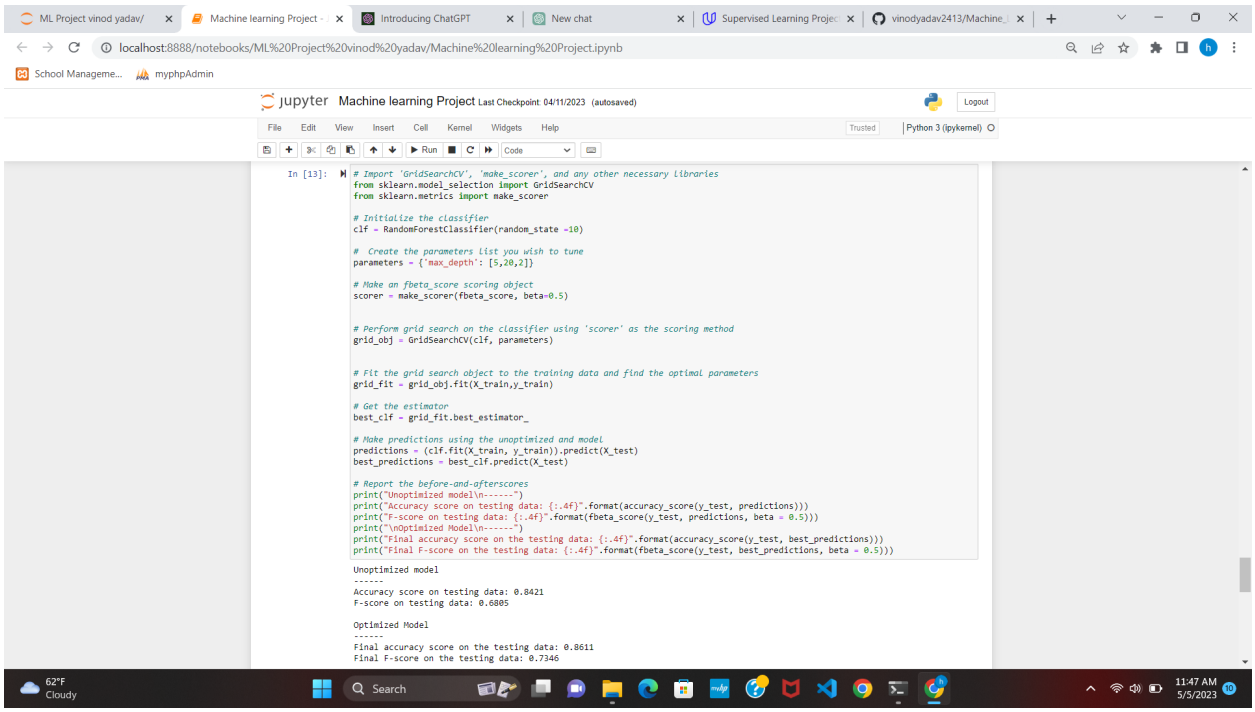
In [12]:
```
# TODO: Import the three supervised learning models from sklearn
```



In [12]:
```
# TODO: Import the three supervised learning models from sklearn
from sklearn import tree
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import AdaBoostClassifier
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier

# Initialize the three models
clf_A = LogisticRegression(random_state=42)
clf_B = AdaBoostClassifier(DecisionTreeClassifier(max_depth=5),random_state=42)
clf_C = svm.SVC(kernel='linear')


# Calculate the number of samples for 1%, 10%, and 100% of the training data
# HINT: samples_100 is the entire training set i.e. len(y_train)
# HINT: samples_10 is 10% of samples_100 (ensure to set the count of the values to be `int` and not `float`)
# HINT: samples_1 is 1% of samples_100 (ensure to set the count of the values to be `int` and not `float`)
samples_1 = int(0.01 * len(X_train))
samples_10 = int(0.1 * len(X_train))
samples_100 = int(len(X_train))

# Collect results on the learners
results = {}
for clf in [clf_A, clf_B, clf_C]:
    clf_name = clf.__class__.__name__
    results[clf_name] = {}
    for i, samples in enumerate([samples_1, samples_10, samples_100]):
        results[clf_name][i] = \
        train_predict(clf, samples, X_train, y_train, X_test, y_test)

# Run metrics visualization for the three supervised learning models chosen
vs.evaluate(results, accuracy, fscore)
```

```
LogisticRegression trained on 361 samples.
```





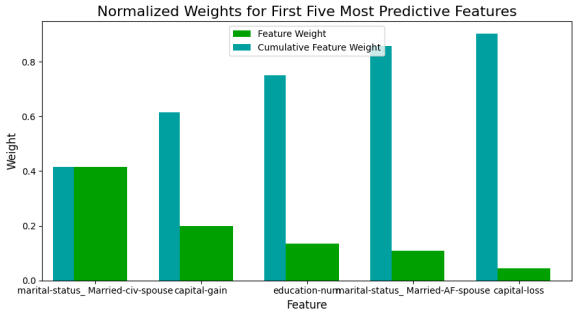
```
In [13]:
# Import 'GridSearchCV', 'make_scorer', and any other necessary libraries
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer

# Initialize the classifier
clf = RandomForestClassifier(random_state =10)

#  Create the parameters list you wish to tune
parameters = {'max_depth': [5,20,2]}

# Make an fbeta_score scoring object
scorer = make_scorer(fbeta_score, beta=0.5)


# Perform grid search on the classifier using 'scorer' as the scoring method
grid_obj = GridSearchCV(clf, parameters)


# Fit the grid search object to the training data and find the optimal parameters
grid_fit = grid_obj.fit(X_train,y_train)

# Get the estimator
best_clf = grid_fit.best_estimator_

# Make predictions using the unoptimized and model
predictions = (clf.fit(X_train, y_train)).predict(X_test)
best_predictions = best_clf.predict(X_test)

# Report the before-and-afterscores
print("Unoptimized model\n------")
print("Accuracy score on testing data: {:.4f}".format(accuracy_score(y_test, predictions)))
print("F-score on testing data: {:.4f}".format(fbeta_score(y_test, predictions, beta = 0.5)))
print("\nOptimized Model\n------")
print("Final accuracy score on the testing data: {:.4f}".format(accuracy_score(y_test, best_predictions)))
print("Final F-score on the testing data: {:.4f}".format(fbeta_score(y_test, best_predictions, beta = 0.5)))
```

```
Unoptimized model
------
Accuracy score on testing data: 0.8421
F-score on testing data: 0.6805

Optimized Model
------
Final accuracy score on the testing data: 0.8611
Final F-score on the testing data: 0.7346
```

```
Optimized Model
------
Final accuracy score on the testing data: 0.8611
Final F-score on the testing data: 0.7346
```

In [14]:

```
# Import a supervised learning model that has 'feature_importances_'

from sklearn.ensemble import GradientBoostingClassifier
# Train the supervised model on the training set using .fit(X_train, y_train)
model = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=1, random_state=0).fit(X_train, y_train)

# Extract the feature importances using .feature_importances_
importances = model.feature_importances_

# Plot
vs.feature_plot(importances, X_train, y_train)
```



In [15]:

```
# Import functionality for cloning a model
from sklearn.base import clone

# Reduce the feature space
X_train_reduced = X_train[X_train.columns.values[(np.argsort(importances)[::-1])[:5]]]
X_test_reduced = X_test[X_test.columns.values[(np.argsort(importances)[::-1])[:5]]]

# Train on the "best" model found from grid search earlier
clf = (clone(best_clf)).fit(X_train_reduced, y_train)

# Make new predictions
reduced_predictions = clf.predict(X_test_reduced)

# Report scores from the final model using both versions of data
print("Final Model trained on full data\n------")
print("Accuracy on testing data: {:.4f}".format(accuracy_score(y_test, best_predictions)))
print("F-score on testing data: {:.4f}".format(fbeta_score(y_test, best_predictions, beta = 0.5)))
print("\nFinal Model trained on reduced data\n------")
print("Accuracy on testing data: {:.4f}".format(accuracy_score(y_test, reduced_predictions)))
print("F-score on testing data: {:.4f}".format(fbeta_score(y_test, reduced_predictions, beta = 0.5)))
```

```
Final Model trained on full data
------
Accuracy on testing data: 0.8611
F-score on testing data: 0.7346

Final Model trained on reduced data
------
Accuracy on testing data: 0.8568
F-score on testing data: 0.7286
```

In [ ]: