Linear Regression and Decision Trees Assignment

Linear Regression Dataset: https://archive.ics.uci.edu/dataset/29/computer+hardware

Decision tree Dataset: https://archive.ics.uci.edu/dataset/105/congressional+voting+records

1.Linear Regression

1.1 Objective

The objective of this data mining prediction is to use linear regression to predict the variable ('199') using the other features that are present in the 'machine.data' dataset. The predictions gathered from this linear regression analysis could be a vital tool for a business to optimize performance of devices, allocate resources, support important decisions, help with financial planning, and predict maintenance for devices. Overall, the results gathered from this linear regression model are a potential asset for an organization to make informed decisions in many different areas.

1.2 Data Exploration

At the beginning of this linear regression analysis all imports where applied to the program

```
import pandas as pd

from sklearn.linear_model import LinearRegression

from pandas.plotting import scatter_matrix

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error
```

The dataset was then loaded from my data folder using the file read and print commands

```
df = pd.read_csv("data/machine.data")

print(type(df))

df.head()
```

To generate the correlation matrix for the dataset the numeric columns 'adviser' and '32/60' were dropped

```
temp =df.drop(["adviser", "32/60"], axis='columns')

temp.corr()
```

In [19]:

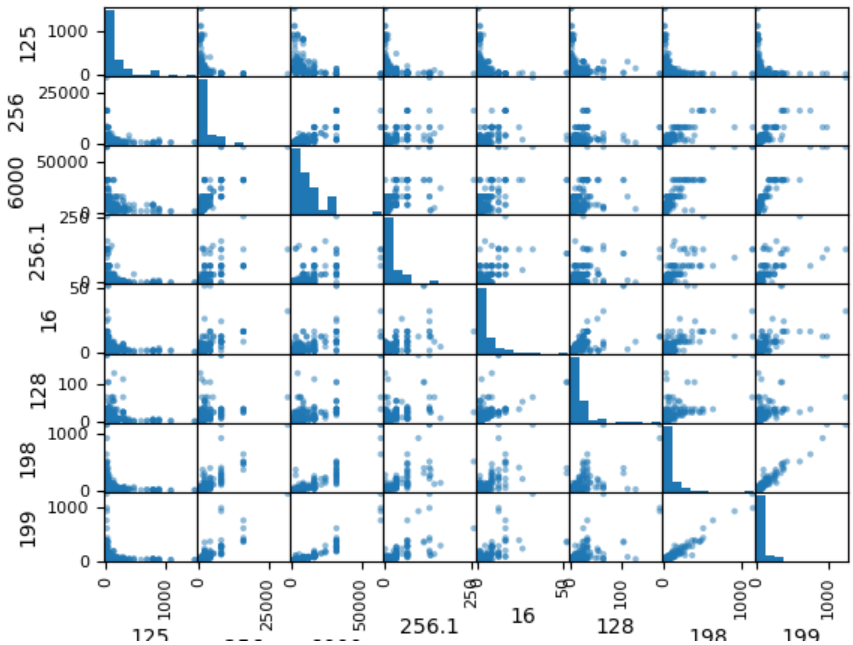
```
temp =df.drop(["adviser", "32/60"], axis='columns')
temp.corr()
```

Out[19]:

	125	256	6000	256.1	16	128	198	199
125	1.000000	-0.337071	-0.379592	-0.340414	-0.300734	-0.255629	-0.306571	-0.287806
256	-0.337071	1.000000	0.757827	0.602788	0.526665	0.293877	0.798310	0.823113
6000	-0.379592	0.757827	1.000000	0.600680	0.568594	0.562388	0.865576	0.904180
256.1	-0.340414	0.602788	0.600680	1.000000	0.588128	0.423550	0.704642	0.687428
16	-0.300734	0.526665	0.568594	0.588128	1.000000	0.541762	0.608841	0.610094
128	-0.255629	0.293877	0.562388	0.423550	0.541762	1.000000	0.621309	0.606281
198	-0.306571	0.798310	0.865576	0.704642	0.608841	0.621309	1.000000	0.966423
199	-0.287806	0.823113	0.904180	0.687428	0.610094	0.606281	0.966423	1.000000

low correlation between 125 and the other variables

I then created scatter matrix plots to visualize the data to further understand the relationships between each variable.



1.3 Modeling

To build the linear regression model, the variable matrix 'X' was created by first removing my target variable '199', and then defining '199' as my target variable.

```
X = df.drop(['adviser','32/60','199'], axis='columns')

y = df['199']
```

The data was then split into separate training and test groups

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)
```

The model then had to be fitted to the training data and the intercept, slopes and R squared values were taken from the model and displayed

```
model = LinearRegression()

model.fit(X_train, y_train)


print('intercept:', model.intercept_)

print('slopes:', model.coef_)

print('R squared:', model.score(X,y))
```

```
intercept: -32.410573316585484
slopes: [ 0.03443313  0.00388484  0.00370053 -0.14183819  0.2572956   0.22162583
  0.64910588]
R squared: 0.9571288178981704
```

The intercept, slopes and R squared value are important metrics when building a model as they provide an insight into the relationship between variables in the model.

1.4 Evaluation

The now trained linear regression model was evaluated using the test set. This was done to generate predictions for the target variable and the model performance results for the mean squared error and the R squared.

```
print(X.adviser.value_counts())

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)

model = LinearRegression()

model.fit(X_train, y_train)

print('R2 :', model.score(X_train, y_train))
```

The revised mean square error

```
yhat = model.predict(X_test)

print('RMSE', mean_squared_error(y_test, yhat, squared=False))
```

1.5 Conclusion

In Conclusion, this model successfully used linear regression to predict the variable (199) based on the other features in the dataset. The model provides efficiency in understanding the relationship between the target variable and other variables. I believe that this model provides a valuable tool for a business to make decisions on strategys to optimize the performance of their computing devices and systems, the model identifies many key factors that optimal device performance is dependent on which would allow for more efficient reasource management and improved overall system performance.

2.Decision Trees

2.1 Objective

The main objective of this data mining prediction is to use decision tree classification to predict a person's political stance based on their voting patterns in the dataset (specifically republican for this decision tree). The decision tree model is designed to be used as a tool for separating people based on their political views which could prove to be a great asset to for future political analysis or polling scenarios where traditional surveys could be insufficient, offering a data-driven approach to understanding political views.

2.2 Data Exploration

First the dataset was loaded and displayed using the read csv function

```
df = pd.read_csv("data/house-votes-84.data")
```

```
df.head()
```

The data was then prepared for modelling by separating the target variable: Republican(y) and the other variables(X)

```
X = df.drop('republican', axis='columns')
```

```
y = df.republican
```

One-hot coding was then utilized to process the categorical features in the dataset

```
X = pd.get_dummies(X)
```

```
feature_names = X.columns
```

```
X.head()
```

	n_?	n_n	n_y	y_?	y_n	y_y	n.1_?	n.1_n	n.1_y	y.1_?	...	y.6_y	y.7_?	y.7_n	y.7_y	n.5_?	n.5_n	n.5_y	y.8_?	y.8_n	y.8_y
0	False	True	False	False	False	True	False	True	False	False	...	True	False	False	True	False	True	False	True	False	False
1	True	False	False	False	False	True	False	False	True	True	...	True	False	False	True	False	True	False	False	True	False
2	False	True	False	False	False	True	False	False	True	False	...	True	False	True	False	False	True	False	False	False	True
3	False	False	True	False	False	True	False	False	True	False	...	True	False	False	True	False	False	True	False	False	True
4	False	True	False	False	False	True	False	False	True	False	...	True	False	False	True	False	False	True	False	False	True

2.3 Modeling

To model the dataset firstly it must be split into training and test sets like the linear regression model before.

```
X_train, X_test, y_train, y_test = train_test_split( X, y,
                                    stratify=y, random_state=1)
```

A decision tree classifier was them used with depths from 1-11 and its performance measured through cross validation.

```
for d in range(1,12):
    model = DecisionTreeClassifier(max_depth=d)
    scores = cross_val_score(model, X_train, y_train, cv=5)
    print("Depth: ", d, "Validation Accuracy:", scores.mean())
```

The best depth for the model was then selected using the results from the cross validation. The decision tree model was then trained on the training data and its accuracy was assessed based on the results from the training and test sets.

```
model = DecisionTreeClassifier(max_depth=7)
model.fit(X_train,y_train)
print("Training Accuracy:", model.score(X_train, y_train))
print("Test Accuracy:", model.score(X_test, y_test))
```

```
Training Accuracy: 1.0
Test Accuracy: 0.944954128440367
```

2.4 Evaluation

Finally to evaluate the model, predictions for the test set were generated and a confusion matrix was generated.

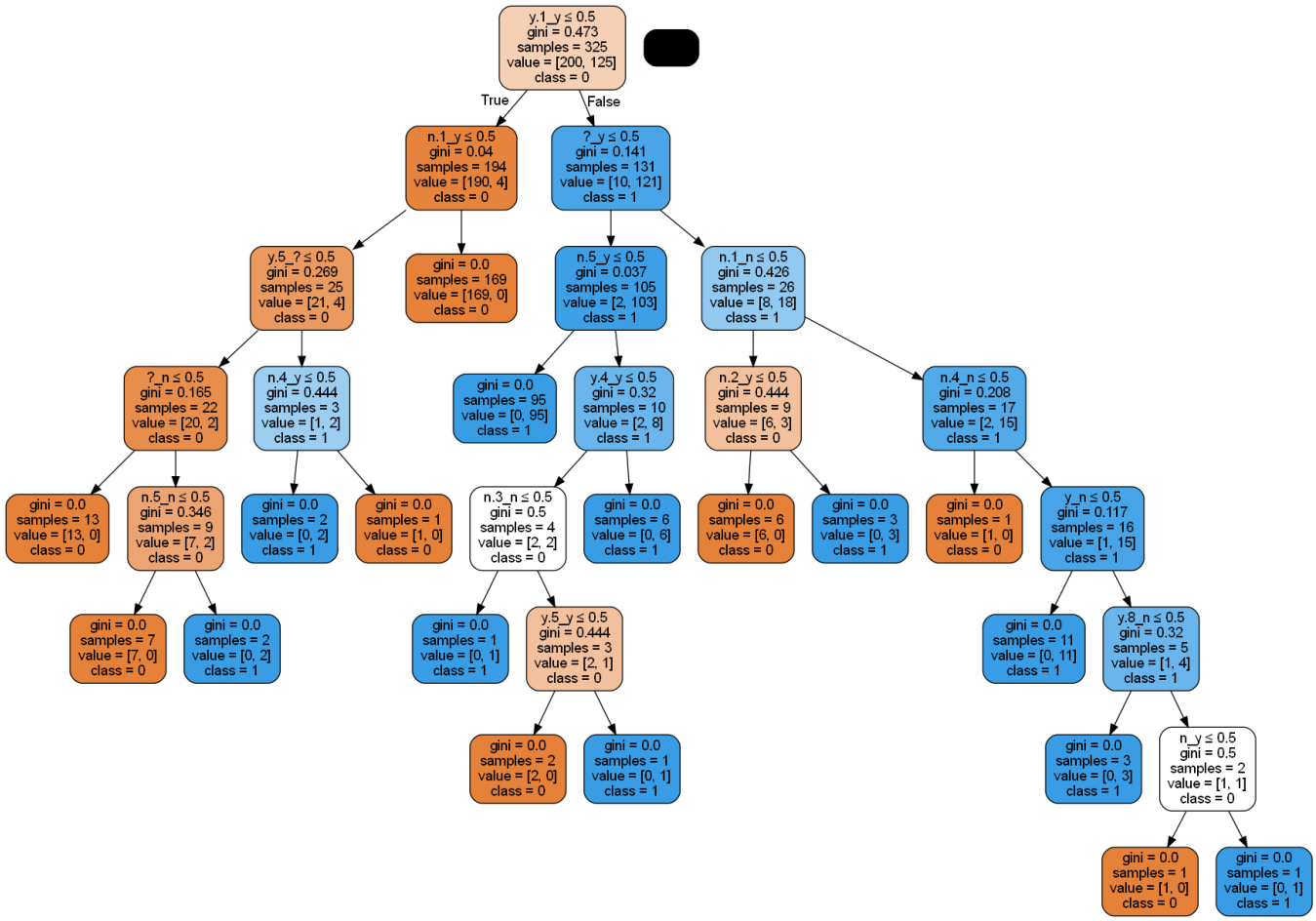
```
y_hat = model.predict(X_test)
cm = confusion_matrix(y_test, y_hat)
print("CM", cm)
print()
tn, fp, fn, tp = cm.ravel()
print("TN", tn, "FP", fp, "FN", fn, "TP", tp)
```

```
CM [[61  6]
 [ 0 42]]

TN 61 FP 6 FN 0 TP 42
```

The above matrix provides values for true positives, true negatives, false negatives, and false positives. The final decision tree was then generated and saved to my 'plots' folder.

```
dot_data = StringIO()
export_graphviz(model, out_file=dot_data,
                filled=True, rounded=True,
               special_characters=True, class_names=['0','1'],
                feature_names = feature_names)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('plots/house.png')
Image(graph.create_png())
```



2.5 Conclusion

The decision tree model was successful in predicting the political views based on voting patterns of people from the data. The model could prove to be particularly useful for political analysts providing a way to visualize voting data and understand voting behavior. I believe the model's accuracy on the training and test data shows its reliability and ability to be used as a tool in the future to make more accurate decisions.