Taylor Schneider - 28026050

CSE5ML: Machine Learning – Assignment 1

Semester 2

Written Report:

1.

To start off, I loaded the dataset into the environment and had a look at the first 6 rows to ensure it was imported correctly. From there it was important to check a few things before getting into cleaning the data.

Firstly I decided to check the structure of the dataset and called the summary to see the top rows and columns. Before we did any work on the data there were 26,215 rows and 10 columns. I uploaded the length of the document as another cross check to ensure it was uploaded correctly and to measure the length of the dataset prior to preprocessing the data. Although there was a small number of columns, from viewing the first 6 rows I could tell there would be a large number of columns added to account for the categorical and ordinal datatypes.

This then leads us on to the next step where I wanted to see the datatypes of each column in the dataset. I checked this by using the dataset.describe function to call the nominal columns. There were only 3 nominal variables out of the 10 columns.

The next two steps were the most generic ways of cleaning the data. First step was to remove any missing values in the data with the ‘na’ function. This was done by calling all the missing values in each column and finding the sum of the total number of missing values in each column. This was called and easily shown which columns have the most missing values. Following the .isna function I used the dropna function to remove the rows with missing values. I chose to remove the rows with missing values because the amount of rows was quite substantial, and there were only two columns with missing values, so I felt this would not skew the data much if they were to be removed.

After removing the missing values it was important to remove any duplicate inputs within the data. I did this with the dataset.drop function and then called the dataset length to see how it had changed.

From here, I then needed to handle the categorical variable and transform them into nominal datatypes so they could be easily computed by the functions within both the Linear Regression model and the SVM model. I was able to see which columns had categorical datatypes from when I previously printed the length of the dataset. After calling the categorical datatypes, I applied ordinal datatype to education, binary to the sex, and dummy data to the remaining categorical columns.

After the data was in the correct format, I was able to split the data into 10 sections and separate 1 section (10%) of the data to use for testing while the remaining 90% of the data was to be used for training the model.

Following this, I normalized the data with a minmax scaler to stabilize the results of outputs when training the data. After normalizing the data we are now ready to create models, train, and test.

2.

Support Vector Machine is one of the most fundamental supervised machine learning techniques and is used primarily for classification problems. THey are simple and elegant when classifying. In SVM, the data, or points are plotted and then separated by a line in 2D instances or a plane in 3D instances to create two categories. SVM technique tries to find a plane/line that categorizes the two group best. Categorisation is done by creating the largest 'margin' between the two groups. This is found with training data and a convex optimization problem. Once the model is trained is it then tested with testing data and finetuned through default parameters and/or intentional parameters set. In the instance where two groups cannot be classified with one SVM, there may need to be another dimension added which then helps categorise the two groups more effectively.

Linear Regression is optimal for predicition problems. Simply put, Linear Regression uses X variables to predict Y. This model works by using a training data set with the summary of information from X and Y to then determine the relation or function between the two. This Function then allows us to receive unknown input data, process through the estimated function of the model and then recieve a accurate prediction of what the outcome would be. Alpha and Beta are the key values in this function.

Both models has great testing accuracy prior to cross validation. However, after receiving the average of test results for each model type it appears the Linear Regression Model is slightly more accurate than SVM making it the better option to use in this instance.

This is apparent in results across all tests, as the linear regression model is just slightly more accurate than the SVM across each result.

<u>Linear Regression Parameters:</u>
'penalty': ['l1', 'l2']
'solver': ['saga']

When choosing saga I wasnt sure if the data set was categorised as a large or small dataset. Generally liblinear is better for small datasets so I tried this first, then saga and saga had a slightly more accurate prediction. L1 & L2 normalisation were chosen because they were synchronised with ‘saga’ and they ensure the penalty doesn’t increase to much to overfit the model.

SVM Parameters:
'kernel': ['linear']

Kernel was used to take into consideration all the inputs (X) in the dataset. Kernel is used because it evaluated the original data and its interactions with its surrounding data as well which then often givens a curved and more accurate decision boundary. It resulted in a slightly more accurate model for SVM results.

For both models the parameters chosen increased the accuracy of the output slightly. With more knowledge I amsure the accuracy of both models could be improved even more.

	Linear Regression	SVM
Testing Accuracy	0.7390900649953575	0.72330547818013
Cross-Val Score	0.7480144308582829	0.7347357728047625
Fine-tuned Accuracy	0.7481807087265839	0.7217142452648929
Test Set Accuracy	0.7395543175487466	0.713091922005571

3.

K means clustering is another machine learning technique that starts with the algorith picking random points within the data based on the number of clusters you have told it to track . From there is basically used those clusters and the input data to train the model and then find new centroids (central points in each cluster) to define the weights used to classify each input and then create a model that can accurately classify any input data.

I chose 2 clusters for my clustering because we only need to know if someone makes above or below $50,000. {0: 9945, 1: 11592} There were 9,945 data samples in one cluster which represented the person making under $50,000 and then 11,592 in the second cluster which represents a person making more than $50,000.

The prototypes extracted from the dataset differ mainly in age the the type of company that person is working for. Each prototype represents one cluster so it shows the accuracy of the model and how the prototypes effectively demonstrate each clustering.

Although the prototypes are accurate, the accuracy from this model is significantly lower than Linear Regression and SVM at 0.4618099. Therefore this model would not be the most effective way to evaluate this data set and Linear Regression is still proven to be the most accurate method.

Code:

```
#!/usr/bin/env python
# coding: utf-8

# In[2]:


import pandas as pd

dataset = pd.read_csv("income.csv")
print("dataset length:", len(dataset))
dataset.head()


# In[88]:


#1a Describe the dataset
dataset.shape
dataset.describe()


# In[17]:


#1b Deal with missing values (if there are any) and use a proper method to handle categorical variables.
#because the amount of na vlues in each row is quite significiant i felt it was best to remove them
```

```
dataset.isna().sum()
dataset = dataset.dropna()
print("dataset length:", len(dataset))


# In[5]:


#1c Remove duplicated inputs if there are any.
dataset.duplicated().any()
dataset = dataset.drop_duplicates()
print("dataset length:", len(dataset))


# In[20]:


#1d Handle the categorical variables.
dataset.dtypes
print(dataset["workclass"].value_counts(),"\n")
print(dataset["education"].value_counts(),"\n")
print(dataset["marital-status"].value_counts(),"\n")
print(dataset["occupation"].value_counts(),"\n")
print(dataset["relationship"].value_counts(),"\n")
print(dataset["race"].value_counts(),"\n")
print(dataset["sex"].value_counts(),"\n")


# In[23]:


#education
dataset['education'] =
dataset['education'].replace({'Preschool':1,'1st-4th':2,'5th-6th':3,'12th':4,'9th':5,'Doctorate':6,'7th-8
th':7,'Prof-school':8, '10th':9,'11th':10,'Assoc-acdm':11,'Assoc-voc':12,'Masters':13,'Bachelors':14,
'Some-college':15,'HS-grad':16})

#sex
dataset['sex']=dataset['sex'].replace({'Male':0, 'Female':1})
```

```
#remaining
dataset = pd.get_dummies(dataset, columns=['race', 'relationship',
'occupation','marital-status','workclass'])
dataset.head()


# In[93]:


#1e split the dataset with 10% for testing
dataset.shape

array=dataset.values
x=array[:,1:14]
y=array[:,0]

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.1, random_state=1)


# In[84]:


#1f
#normalization
from sklearn.preprocessing import MinMaxScaler

norm = MinMaxScaler().fit(x_train)
x_train_norm = norm.transform(x_train)
x_test_norm = norm.transform(x_test)


# In[94]:


#2a
#define Regression & SVM
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
```

```
# Logistic regression model for prediction
model = LogisticRegression(solver="liblinear")
model.fit(x_train, y_train)
test_score = model.score(x_test, y_test)
print("Testing Accuracy of LR:", test_score)

# Support Vector Machine for classification
model = SVC()
model.fit(x_train, y_train)
test_score = model.score(x_test, y_test)
print("Testing Accuracy of SVC:", test_score)


# In[82]:


#2b
#define 10-fold cross validation
from sklearn.model_selection import KFold
kfold=(KFold(n_splits=10, shuffle=True, random_state=1))

from sklearn.model_selection import cross_val_score

model = LogisticRegression(solver="liblinear")
results = cross_val_score(model, x, y, cv=kfold)
print('Average Accuracy of LR:', results.mean())

model = SVC()
results = cross_val_score(model, x, y, cv=kfold)
print('Average Accuracy of SVM:', results.mean())


# In[99]:


#2c Fine tune parameters for LG model
from sklearn.model_selection import GridSearchCV

grid_params_lr = {
    'penalty': ['l1', 'l2'],
```

```
    'solver': ['saga']
}

lr = LogisticRegression(max_iter=150)
gs_lr_results = GridSearchCV(lr, grid_params_lr, cv=kfold).fit(x_train_norm, y_train)
print(gs_lr_results.best_score_)


# In[105]:


#2c fine tune parameters for SVM model
from sklearn.model_selection import GridSearchCV

grid_params_svc = {
'kernel': ['linear'],
}

svc = SVC()
gs_svc_result = GridSearchCV(svc, grid_params_svc, cv=kfold).fit(x_train_norm, y_train)
print(gs_svc_result.best_score_)


# In[107]:


#2d LG evaluate trained model
test_accuracy = gs_lr_results.best_estimator_.score(x_test_norm, y_test)
print("Accuracy in testing:", test_accuracy)


# In[109]:


#2d SVM evaluate trained model
test_accuracy = gs_svc_result.best_estimator_.score(x_test_norm, y_test)
print("Accuracy in testing:", test_accuracy)


# In[126]:
```

```
#3a apply clustering to normalized data
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2, random_state=1).fit(x)

print('kmeans.cluster_centers_:',kmeans.cluster_centers_)


# In[116]:


#3b identify how data samples in each cluster
kmeans_labels = kmeans.labels_
unique_labels, unique_counts = np.unique(kmeans_labels, return_counts=True)
dict(zip(unique_labels, unique_counts))


# In[118]:


#3c Extract a prototype frome each cluster
from sklearn.metrics.pairwise import pairwise_distances_argmin

kmeans_cluster_centers = kmeans.cluster_centers_
closest = pairwise_distances_argmin(kmeans.cluster_centers_, x)
dataset.iloc[closest, :]


# In[128]:


from sklearn.metrics import accuracy_score

y = array[:,-1]
kmeans_labels = kmeans.labels_
# or kmeans_lables = kmeans.predict(X)

accuracy = accuracy_score(y, kmeans_labels)
```

```
print("k means prediction accuracy:", accuracy)
```