

Homework 4 (100 points) (Regression & Classification)

This assignment has 3 separate problems. Each problem will require you to create a separate source code pipeline (using a jupyter notebook) and perform all the requisite analyses. In addition to performing the analysis (i.e., writing code in the jupyter notebook), any questions that have been posed in the word document can be answered directly inline (i.e., below each question in the document itself). The assignment is worth 100 points in total.

What to submit:

1. For each question, you will need to create a separate jupyter notebook with all the requisite code / analysis.
2. In addition, you can use this word document to answer all the questions (inline) and submit a single document for all the analysis.

Submission format: Do not modify the directory structure of the zip archive that you download to start the assignment. Essentially, make sure each jupyter notebook is in a separate directory (which is how the assignment is initially provided i.e., with Q1, Q2, Q3 subdirectories). The *HW4_Questions_and_solutions.pdf* (i.e., the current file with the solutions that you will fill in and convert to PDF format), should be placed at the root of the zip archive.

Once all the solution pdf and all the source code is complete, zip the full assignment and submit the zip archive. Your submission zip archive should have the following structure:

```
<lastname_firstname_CS556B_HW4.zip>/
  /HW4_Questions_and_Solutions.pdf
  Q1/q1.ipynb
  Q2/q2.ipynb
  Q3/q3.ipynb
```

Leave the datasets unchanged and in the directory itself. Each of your jupyter notebooks should be directly executable without any further modifications (i.e., if your assignment submission is downloaded from canvas and unzipped), any of your submitted notebooks if executed using a jupyter server should be directly runnable without additional data downloads etc.

Q1. Housing Price Prediction (34 points)

- a. **Dataset Description:** The data pertains to the houses found in each California district and some summary statistics about them based on the 1990 census data. It contains one instance per district block group. A block group is the smallest geographical unit for which the U.S. Census Bureau publishes sample data (a block group typically has a population of 600 to 3,000 people).

The goal of this task is to design a regression model to predict the *median house value* conditioned upon a set of input attributes corresponding to a particular California district block.

The attributes in the dataset are as follows; their names are self-explanatory:

- i. longitude (continuous): One of the coordinates that are used to identify the California district block
- ii. latitude (continuous): One of the coordinates that are used to identify the California district block
- iii. housing_median_age (continuous): Average age of the house in California district block
- iv. total_rooms (continuous): Total number of rooms of all the houses in the California district block
- v. total_bedrooms (continuous): Total number of bedrooms of all the houses in the California district block
- vi. population (continuous): Number of people residing in the district block
- vii. households (continuous): Number of families in the district block
- viii. median_income (continuous): Median income for households in the district block of houses (measured in tens of thousands of US Dollars)
- ix. ocean_proximity (categorical): Location of the house. Is it inland, near the bay, near the ocean, etc.
- x. median_house_value (continuous): Median house value within a district block (measured in US Dollars)

Our target variable will be median_house_value. Use the rest of the fields mentioned above to predict the median_house_value.

b. **Data Loading / Preprocessing: (10 points)**

- i. Loading: (2 points)
 1. Load the California housing dataset using `pandas.read_csv()` function and store it in the variable (i.e., a pandas dataframe) named 'df'.
 2. The resulting data frame should have the shape (20640, 10) indicating that there are 20640 rows and 10 columns.
 3. Find the missing values in the data frame. If any (i.e., even if one column in each instance / row has a missing value), drop the row using `pandas.DataFrame.dropna()` function. The resulting data frame should have the shape (20433, 10) indicating that there are 20433 rows and 10 columns.
 4. Create a data frame 'corr_df' by dropping the columns *latitude*, *longitude*, and *ocean_proximity* using the `pandas.DataFrame.drop()` function. Use the *Pearson correlation* to find the correlation of each remaining feature in the 'corr_df' with the target variable '*median_house_value*' using the function `pandas.DataFrame.corrwith()`. Report results in the following table.

Feature Name	Pearson Correlation
housing_median_age	0.106432
total_rooms	0.133294
total_bedrooms	0.049686
population	-0.025300
households	0.064894
median_income	0.688355

Commented [MN1]: The explanation is confusing. Part of the description talks about a "district block group" but the other half talks about the entire district. Which one is it?

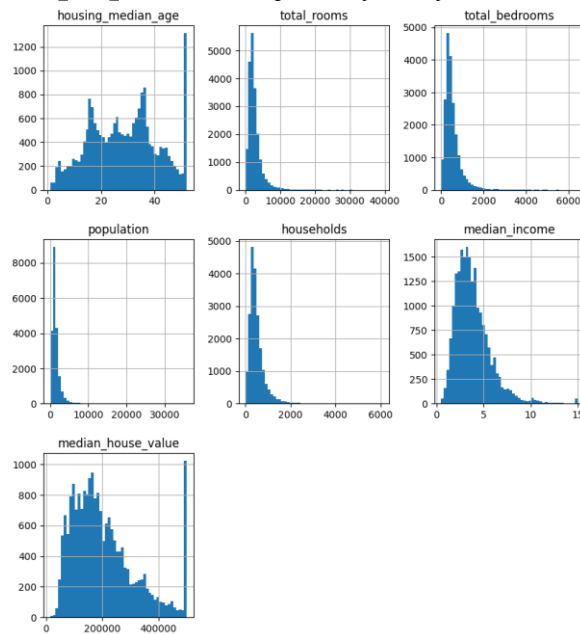
Commented [NM2]: change the name of this to something else. 'temp' is usually not very descriptive.

5. Create a data frame X of features (by dropping the column 'median_house_value' from the original data frame) using the `pandas.DataFrame.drop()` function. Create a Series object of targets Y (by only considering the 'median_house_value' column from the original data frame (Do NOT use the 'corr_df' data frame in this step. Use the data frame which was obtained as a result of step b.i.3 above).

Commented [NM3]: again update this name to something else as you will change the name 'temp'

ii. Data Visualization: (3 points)

1. Use `pandas.DataFrame.hist(bins = 50)` function for visualizing the variation on the columns *housing_median_age*, *total_rooms*, *total_bedrooms*, *population*, *household*, *median_income* and *median_house_value*. Plot each histogram as a separate subplot.



2. Use `pandas.dataframe.describe()` function to find the mean, median and standard deviations for each feature and report in the jupyter notebook.
3. Use `pandas.get_dummies` to convert categorical variables into dummy / one-hot encoding. In this case the categorical column is *ocean_proximity*

Commented [NM4]: nice this is good.

iii. Data Splitting: (2 points)

1. Split data into training and test sets using the sklearn `train_test_split()` function. Perform 70-30 distribution i.e. 70% training and 30% testing. The result of your data split should yield 4 separate data frames *X_train*, *X_test*, *y_train*, *y_test*. (respectively, the training features, testing features, training targets and testing target).

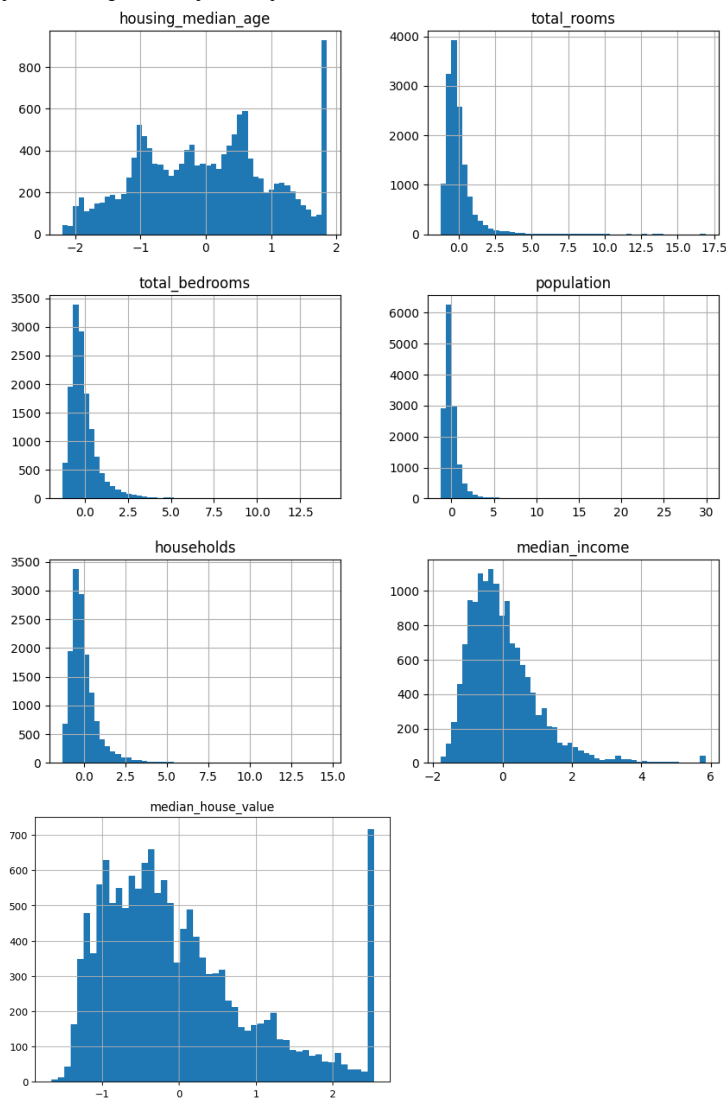
iv. Data Scaling: (3 points)

1. Use the `StandardScaler()` to instantiate the standard scaler class. **Note:** You will need two separate scaler objects, one to scale the features, another to scale the target values.
2. For each scaler, employ the `'fit_transform()'` function (only on the training features, training targets) of the scaler to retrieve the new (scaled) version of the data. Store them in *X_train*, and

`y_train` again.

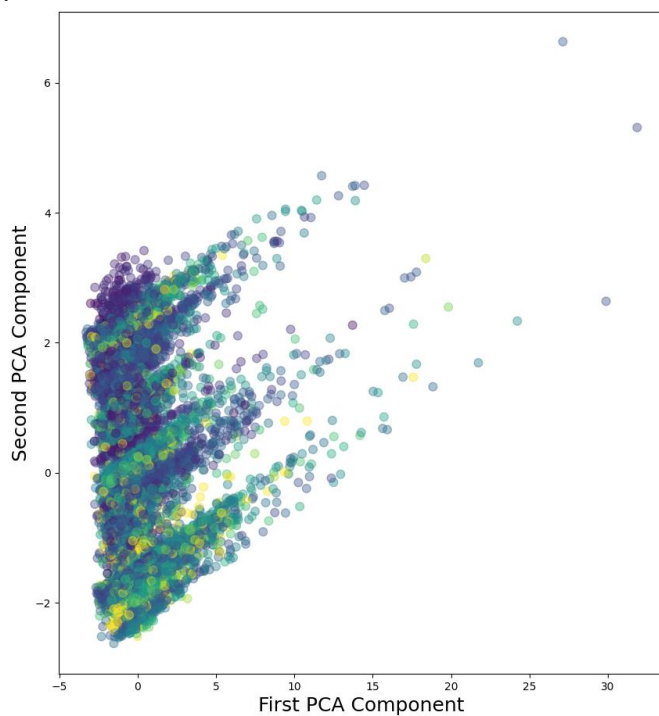
3. Scale the X_{test} and y_{test} as well and store the scaled values back in X_{test} and y_{test} . (i.e., use the appropriate “fitted” scaler above to “transform” the test data. Note: the function to be employed in this case is `transform()` as opposed to `fit_transform()`). Henceforth, X_{train} , y_{train} , X_{test} , y_{test} will refer to the scaled data unless stated otherwise.
4. Use `pandas.DataFrame.hist(bins = 50)` function for visualizing the variation of numerical attributes `housing_median_age`, `total_rooms`, `total_bedrooms`, `population`, `household`, `median_income` and `median_house_value` for the X_{train} and y_{train} dataset (similar to step b.ii.1 above). Once again, plot each histogram as a separate subplot.

Commented [NM5]: please list which ones these are explicitly.



c. **Modelling: (10 points)**

- i. Employ Linear Regression from [sklearn.linear_model](#), and instantiate the model.
- ii. (3 points) Once instantiated, 'fit()' the model using the *scaled* X_{train} , y_{train} data.
- iii. (2 points) Employ the 'predict()' function to obtain predictions on X_{test} . Store the predictions in a variable named 'y_preds'. Note: Since the model has been trained on scaled data (i.e., both features and targets, the predictions will also be in the "scaled" space. We need to transform the predictions back to the original space).
- iv. (2 points) Use [inverse_transform\(\)](#) function to convert the normalized data ('y_preds') to original scale. Store the transformed values back into 'y_preds'.
- v. (3 points) Perform [PCA](#) on the features (X_{train}) and set $n_component$ as 2.
 1. Show a scatter plot where on the x-axis we plot the first PCA component and second component on the y-axis.

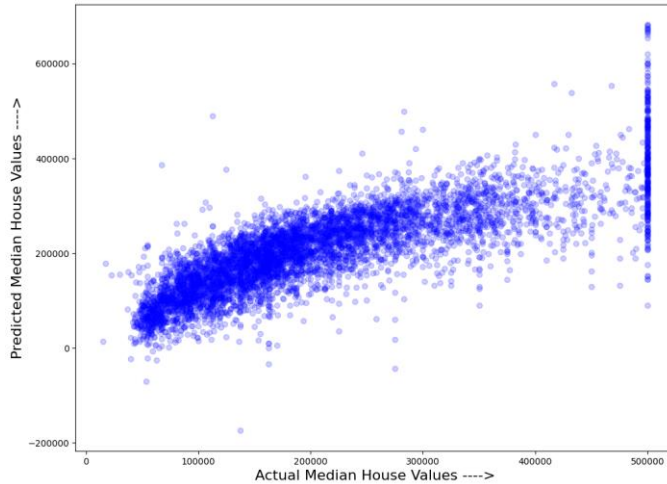


2. Calculate the total percentage of variance captured by the 2 PCA components using [pca.explained_variance_ratio](#). Also, report the strength of each PCA component using [pca.singular_values](#).

```
Explained variation per principal component: [0.30110561 0.19742525]
Total percentage of variance captured by the first 2 components of PCA: 49.85308586784514
Strength of each PCA components: [236.61630546 191.59606813]
```

d. **Evaluation: (6 points)**

- i. (2 points) Plot a scatter plot using [matplotlib.pyplot.scatter](#) function. Plot the predicted median house values on the y-axis vs the actual median house values on the x-axis.



- ii. (4 points) Calculate [MAPE](#), [RMSE](#) and [R²](#) for the model and report them in the following table.
Hint for RMSE set the *squared* parameter to False.

Model	MAPE	RMSE	R ²
Linear Regression	0.2875296239403572	222.8069258299707	0.6538727132608757

c. Discussion: (8 points)

- i. (2 points) Based on the weights of the linear regression model, rank the features (note: only continuous features) in decreasing order of influence on the predictions.

Ans i: Based on the weights of the linear regression model, rank of features are given below:

Rank 1: median_income = 0.649
 Rank 2: total_bedrooms = 0.388
 Rank 3: households = 0.125
 Rank 4: housing_median_age = 0.12
 Rank 5: total_rooms = -0.124
 Rank 6: population = -0.349
 Rank 7: longitude = -0.465
 Rank 8: latitude = -0.472

- ii. (2 points) Discuss how the influence of the features (obtained in question e.1) relates to the pair-wise correlation results calculated above i.e., are the features that are highly correlated with the output also the most influential or is there some other phenomenon being observed?

Ans ii: Correlation table ranks:

Rank 1: median_income = 0.688
 Rank 2: total_rooms = 0.133
 Rank 3: housing_median_age = 0.106
 Rank 4: households = 0.065
 Rank 5: total_bedrooms = 0.05
 Rank 6: population = -0.025

The phenomenon observed here is that both case have feature median_income high rank and feature population low rank. If we doesn't consider the 1st case both Rank 7 and Rank 8 features i.e., longitude and latitude. Then we see that 1st case feature total_bedrooms which is Rank 2 becomes 2nd last Rank i.e., Rank 5 for 2nd case. Then again we see that 1st case feature households which is Rank 3 becomes 3rd last Rank i.e., Rank 4 for 2nd case. Then again we see that 1st case feature housing_median_age which is Rank 4

becomes 4th last Rank i.e., Rank 3 for 2nd case. Then we see that 1st case feature total_rooms which is Rank 5 becomes 5th last Rank i.e., Rank 2 for 2nd case. This pattern is observed while comparing both cases.

- iii. (2 points) Comment about the MAPE, RMSE, R^2 results. What can we learn from each of these results about the model prediction performance?

Ans iii: As the value of MAPE 0.29 (approx.) is very close to zero, this indicates that we have a good model prediction. The percentage error is almost zero.

The value of RMSE 222.81 (approx.) is high, we can say the model prediction performance is not that good. The value of R^2 0.65 (approx.) is close to 1, we can say that the model prediction performance is working well (though it is not true for every case).

- iv. (2 points) Why is centering and scaling the data important before performing PCA?

Ans iv: PCA tries to get the features with the maximum variance and the variance is high for high magnitude features. This skews the PCA towards high magnitude features, which might lead to improper selection of principle components. That's why we require scaling the data before applying PCA. Principal components analysis (PCA) is based conventionally on the eigenvector decomposition (EVD). Mean-centering the input data prior to the eigen-analysis is treated as an integral part of the algorithm. It ensures that the first principal component is proportional to the maximum variance of the input data. That's why we require centering the data before applying PCA.

Q2. Titanic Classification Problem (33 points)

- a. **Dataset Description:** The sinking of the Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the widely considered “unsinkable” RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren’t enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew. While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others. Our goal is to develop a classifier to predict whether a passenger *survived* the calamity. The attributes in the datasets are as follows:

- i. PassengerID (categorical): Passenger ID
- ii. Ticket (categorical): Ticket Number
- iii. Name (categorical): Passenger Name
- iv. Cabin (categorical): Passenger Cabin
- v. Pclass (categorical): Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
- vi. Sex (categorical): Passenger Sex
- vii. Age (continuous): Passenger Age
- viii. SibSp (continuous): Number of Siblings/Spouses Aboard
- ix. Parch (continuous): Number of Parents/Children Aboard
- x. Fare (continuous): Passenger Fare
- xi. Embarked (categorical): Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)
- xii. Survived (categorical): Survival (0 = No; 1 = Yes) --- this is the target column. Make sure to encode data appropriately for each classifier.
- xiii. Our target variable will be Survived. Use the rest of the fields mentioned above to predict whether a passenger survived the Titanic shipwreck.

b. **Data Loading / Preprocessing (10 points)**

- i. Loading (4 points)
 1. Load the data <df_train.csv> and <df_test.csv> as a pandas dataframe using the ‘pandas.read_csv’ function. The ‘df_test.csv’ has been preprocessed (i.e., null values have been dropped, certain columns etc. have been dropped) and should not be changed apart from splitting the the dataframe into X_test and y_test. The ‘df_train’ data has NOT been preprocessed and you will need to preprocess and prepare the ‘df_train’ dataframe. Note: Neither *df_train* nor *df_test* have been scaled. The next few steps will enumerate data preprocessing, scaling requirements we need to perform.
 2. The resulting dataframe (i.e., *df_train*) should have the shape (712,12) indicating that there are 712 instances and 12 columns.
 3. In *df_train*, dataframe, currently you have 12 columns which are the following – PassengerID, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, Embarked and the Survived column (target variable).
 4. Use the ‘pandas.isnull().sum()’ function check if there are any missing values in the df_train dataframe. Report which columns have missing (i.e., null) values and provide the number of the null values in the columns.

Commented [NM6]: this is confusing.

Commented [NM7R6]: Can we not just say none of the dataframes have been scaled?


```

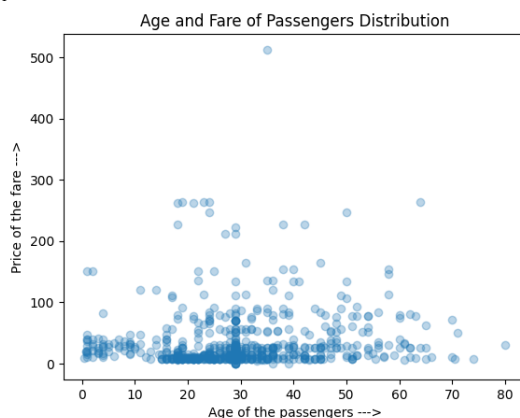
PassengerId    0
Pclass         0
Name           0
Sex            0
Age          138
SibSp          0
Parch          0
Ticket         0
Fare           0
Cabin         541
Embarked       2
Survived       0
dtype: int64

```

5. Use the '`pandas.DataFrame.drop()`' function to drop the 'Cabin', 'PassengerID', 'Name' and 'Ticket' columns.
6. Use the '`pandas.DataFrame.fillna()`' function to replace the NA values in the 'Age' column with the mean value of the 'Age' column. Note: This process is called 'imputation' (i.e., filling null values with a pre-specified value) and we are employing one strategy called *mean imputation*, but other strategies can also be employed in general. Use the '`dropna()`' function to drop any remaining rows that consist of NA values.
7. Your task is to use the feature columns to predict the target column. This can be cast as a classification problem.
8. Create a pandas dataframe `X_train` of features (by dropping the 'Survival' column from the `df_train` dataframe). Create a pandas `Series` object of targets `y_train` (by only considering the 'Survival' column from the `df_train` dataframe). Moving forward, we will be working with `X_train` and `y_train`. At this point also split the `df_test` into `X_test` and `y_test` by dropping the 'Survival' column and storing the features into `X_test`. Store the 'Survival' column in `y_test`.

ii. Data Visualization (4 points)

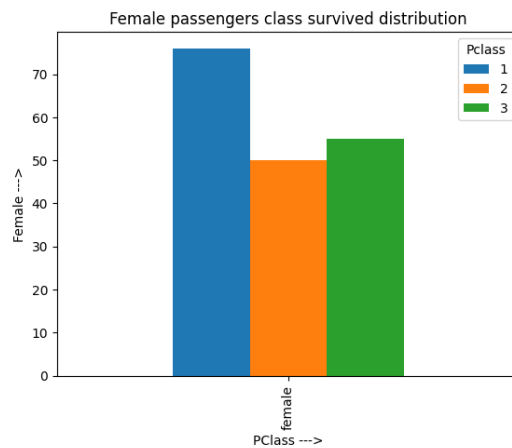
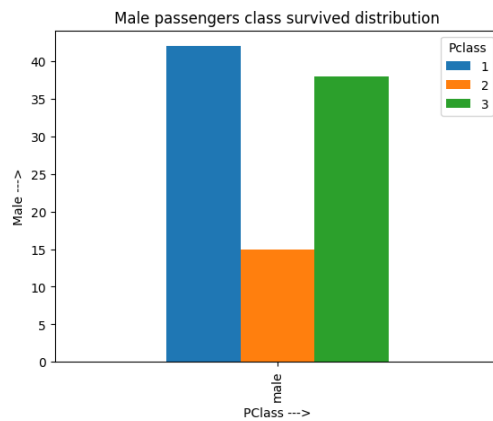
1. Using matplotlib employ a scatter plot using '`matplotlib.pyplot.scatter`' between the age of the passengers and the price of their fare. Label the x-axis and the y-axis along with the giving the plot a title.



- a. What is the highest and lowest fare price?
Ans a: The highest fare price is \$512.33 and lowest fare price is \$0.0

- b. What are the respective mean values of these two features?
 Ans b: The mean value of Age is 29.65 years and mean value of fare is \$32.66
- c. What was the age of the oldest passenger?
 Ans c: The age of oldest passenger is 80 years.

2. **Only for this question** use the `df_train` dataframe. Using matplotlib visualize the number of males and females that survived and their respective *passenger classes* on two separate bar chart plots using '`matplotlib.pyplot.bar`' (Passenger Class column).



- a. Which *class of passengers* had the least number of survived males and how many? Repeat this analysis for females.

Ans a: Class 2 of passengers had the least number of survived males and that is 15.
 Class 2 of passengers had the least number of survived females and that is 50.

- b. Which *class of passengers* had the greatest number of survived males and how many? Repeat this for females.

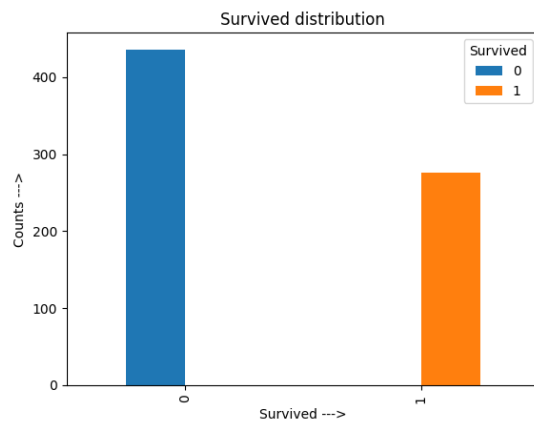
Commented [NM8]: this said 'class' previously and this might be confusing in a classification problem. Hence I change this to 'passenger class' as that is what I am assuming you meant? If so, leave unchanged, if not change appropriately.

Commented [NM9]: same comment as above.

Commented [NM10]: same comment as above.

Ans b: Class 1 of passengers had the greatest number of survived males and that is 42.
Class 1 of passengers had the greatest number of survived females and that is 76.

3. Using the Target variable (Survived) in y_train plot a bar chart showing the distribution of the 'Survived' column.



- a. What initial comment can you make about this distribution in terms of the frequency of each class?

Ans a: Initial comment is it shows that class 0 has more frequency than class 1 which means majority of passengers couldn't survive.

4. So far you should have successfully been able to load, preprocess and visualize your data. Use the 'pd.get_dummies()' function to convert categorical data into dummy variables ('Sex' and 'Embarked'). (Perform this only on X_train store the result back into X_train).
- a. What is the new shape of X?

Ans a: The new shape of X is (712, 8)

iii. Data Scaling (2 points)

- Employ X_train and MinMaxScaler only on the continuous attributes. Employ the 'fit_transform()' function of the scaler to retrieve the new (scaled) version of the data. Store the scaled values in X_train again.
- Scale the X_test using the scaler you have just fit, this time using the 'transform()' function. Note: store the scaled values back into X_test. At the end of this step, you must have X_train, X_test, all scaled according to the MinMaxScaler.

c. Modelling (10 points)

- (2 points) Modelling (Model Instantiation / Training) using **Logistic Regression classifier**
 - Employ the Logistic Regression classifier from sklearn and instantiate the model. Label this model as 'model_lr'
 - Once instantiated, 'fit()' the model using the *scaled* X_train, y_train data.
 - Employ the 'predict()' function to obtain predictions on X_test and store this in a variable labeled as 'y_pred_lr'.

Commented [NM11]: this question is too open ended. Maybe make it slightly more specific and add some details about what you'd like them to comment about.

Commented [NM12]: Specify the columns on which this operation should be performed.

Commented [NM13]: I added this part, is this the intention?

Commented [NM14]: Make sure to mention that ONLY continuous attributes should be scaled as we don't want to scale categorical attributes.

Also, for this reason (i.e., since we have continuous and categorical attributes), let us use minmax scaling.

4. Employ the `accuracy_score` function by using the 'y_pred_lr' and 'y_test' variables as the functions parameters and print the accuracy of the Logistic Regression model.

Commented [NM15]: Check bank churn for similar comment about this function.

ii. (2 points) **Modelling (Model Instantiation / Training) using Support Vector Machine Classifier**

1. Employ the Support Vector Machine (SVM) classifier from sklearn () and instantiate the model. Label this model as 'model_svm'
2. Once instantiated, 'fit()' the model using the *scaled* X_train, y_train data.
3. Employ the 'predict()' function to obtain predictions on X_test and store this in a variable labeled as 'y_pred_svm'.
4. Employ the `accuracy_score` function by using the 'y_pred_svm' and 'y_test' variables as the functions parameters and print the accuracy of the SVM model.

Commented [NM16]: check bank churn for similar comment about this function.

iii. (3 points) **Modelling Logistic Regression Classifier** with the addition of noise on the target variable. In the data repository you should see three noisy datasets – 'df_train_noise20', 'df_train_40', 'df_train_60'. These datasets have already been preprocessed. In each dataset 'df_train_noise<integer>', the integer indicates the percentage of noise injected into the target variable in that training set. The noise can be considered a result of incorrect class labelling of a particular instance. For example, in df_train_noise20, 20% of the instances have an incorrect target label in the training set. Our goal will be to train a set of classification models on such noisy training data and test on a clean test set (i.e., same as what we have been using so far 'df_test'). Load the 'df_train_noise<nl>' (nl means noise level and is a place holder for the integer percentage) datasets and split the data into X_train_<nl> and y_train_<nl> (e.g., if working with 'df_train_noise20' we would split the data and store it in variables named X_train_20, y_train_20). 'X_train_<nl>' should store the features and 'y_train_<nl>' should store the target variable.

Repeat the following steps (1 – 4) for the 20%, 40%, 60% noise level datasets.

Train a new Logistic Regression model on the new training and use the pre-existing X_test and y_test to evaluate your model. Label this model as 'model_lr_noise_<nl>'. Specifically, do the following:

1. Employ a new Logistic Regression classifier from sklearn and instantiate the model. Label this model as 'model_lr_noise<nl>'
2. Once instantiated, 'fit()' the model using the X_train_<nl> and y_train_<nl> data.
3. Employ the 'predict()' function to obtain predictions on X_test and store this in a variable labeled as 'y_pred_lr_noise<nl>'.
4. Employ the `accuracy_score` function and print the accuracy of the new Logistic Regression model.
 - a. What is your initial observation of the accuracy, is the accuracy higher or lower than that of the clean dataset?

Commented [NM17]: same comment as in bank churn.

Ans a: The initial observation of the accuracy is that all the three noisy data's accuracy is lower than that of the clean dataset accuracy.

iv. (3 points) **Modelling Support Vector Machine (SVM) Classifier** with the addition of noise on the target variable. In the data repository you should see three noisy datasets – 'df_train_noise20', 'df_train_40', 'df_train_60'. These datasets have already been preprocessed. In each dataset 'df_train_noise<integer>', the integer indicates the percentage of noise injected into the target variable in that training set. The noise can be considered a result of incorrect class labelling of a particular instance. For example, in df_train_noise20, 20% of the instances have an incorrect target label in the training set. Our goal will be to train a set of classification models on such noisy training data and test on a clean test set (i.e., same as what we have been using so far 'df_test'). Load the 'df_train_noise<nl>' (nl means noise level and is a place holder for the integer percentage) datasets and split the data into X_train_<nl> and y_train_<nl> (e.g., if working with 'df_train_noise20' we would

split the data and store it in variables named X_train_20, y_train_20). 'X_train_<n>' should store the features and 'y_train_<n>' should store the target variable.

Repeat the following steps (1 – 4) for the 20%, 40%, 60% noise datasets.

Train a new SVM classification model on the new training and use the pre-existing X_test and y_test to evaluate your model. Label this model as 'model_svm_noise_<n>'. Specifically, do the following:

1. Employ a new SVM classifier from sklearn and instantiate the model. Label this model as 'model_svm_noise_<n>'
2. Once instantiated, 'fit()' the model using the X_train_<n> and y_train_<n> data.
3. Employ the 'predict()' function to obtain predictions on X_test and store this in a variable labeled as 'y_pred_svm_noise_<n>'.
4. Employ the 'accuracy_score' function and print the accuracy of the new Logistic Regression model.
 - a. What is your initial observation of the accuracy, is the accuracy higher or lower than that of the clean dataset?

Ans a: The initial observation of the accuracy is that all the three noisy data's accuracy is lower than that of the clean dataset accuracy.

d. Evaluation (5 points)

- i. (3 points) Report F1 Score, Precision, Recall, Accuracy (All on the test set X_test, y_test)
 1. Employ a 'classification_report()' function from sklearn.metrics to report the precision recall and f1 score for each class for the 'model_lr' model and the 'model_svm' along with a confusion matrix for each of them.
- ii. (2 points) Report the accuracy and classification report for each of the three noisy models (model_lr_noise_<n>, model_svm_noise_<n>).

e. Discussion (8 points)

- i. (4 points) Compare the performance of the Logistic Regression (model_lr) classifier and the SVM classifier (model_svm) trained on the clean training data sets. Using the classification reports which model performs better? Mention the specific numbers (i.e., from your results calculating the precision, recall, F1 score) in a table.

Ans i:

Model Name	Class	Precision	Recall	F1
model_lr	0	0.90	0.93	0.91
model_lr	1	0.87	0.82	0.84
model_svm	0	0.83	0.98	0.90
model_svm	1	0.96	0.67	0.79

From the above table and classification reports we can conclude that overall performance of Logistic Regression (model_lr) classifier is better than SVM classifier (model_svm) trained on the clean training data sets.

Accuracy of model_lr 0.89 > Accuracy of model_svm 0.87

Class 0 of model_lr Precision 0.90 > Class 0 of model_svm Precision 0.83

Class 0 of model_lr F1 0.91 > Class 0 of model_svm F1 0.90

Class 1 of model_lr Recall 0.82 > Class 1 of model_svm Recall 0.67

Class 1 of model_lr F1 0.84 > Class 1 of model_svm F1 0.79

Class 0 of model_lr Recall 0.93 < Class 0 of model_svm Recall 0.98

Class 1 of model_lr Precision 0.87 < Class 1 of model_svm Precision 0.96

Here only in two cases model_svm metrics score are more than model_lr metrics score but majority the model_lr metrics score are more than model_svm metrics.

Commented [NM18]: same comment as in bank churn.

Commented [NM19]: name the models.

Commented [NM20]: name the models.

- ii. (4 points) Report the performance of all the classification models you have trained thus far in terms of (precision, recall, F1 scores). From the results of the three noisy models implemented using Logistic Regression. What did you notice about the accuracy as the noise of the dataset increased? What can you say about the effect of noise on data mining pipelines?

Ans ii:

Model Name	Class	Precision	Recall	F1
model_lr	Not Survived	0.90	0.93	0.91
model_lr	Survived	0.87	0.82	0.84
model_lr_noise20	Not Survived	0.85	0.92	0.88
model_lr_noise20	Survived	0.84	0.71	0.77
model_lr_noise40	Not Survived	0.74	0.97	0.84
model_lr_noise40	Survived	0.90	0.41	0.56
model_lr_noise60	Not Survived	0.68	0.73	0.70
model_lr_noise60	Survived	0.47	0.41	0.44
model_svm	Not Survived	0.83	0.98	0.90
model_svm	Survived	0.96	0.67	0.79
model_svm_noise20	Not Survived	0.79	1.00	0.88
model_svm_noise20	Survived	1.00	0.55	0.71
model_svm_noise40	Not Survived	0.82	0.98	0.90
model_svm_noise40	Survived	0.95	0.64	0.76
model_svm_noise60	Not Survived	0.67	0.81	0.74
model_svm_noise60	Survived	0.50	0.32	0.39

From the results of the three noisy models implemented using Logistic Regression, we notice that the accuracy is gradually decreased as the dataset noise increased.

And from the results of the three noisy models implemented using Support Vector Machine, we notice that the accuracy is first increased and then decreased as the dataset noise increased.

The effect of noise on data mining pipelines is that it unnecessarily increases the amount of storage space required and can also adversely affect the results of any data mining analysis such as change in accuracy score, increases the data processing time complexity and many more.

Q3. Bank Churn Classification Problem (33 points)

- a. **Dataset Description:** Banking is one of those traditional industries that has gone through a steady transformation over the past few decades. Yet, many banks today with a sizeable customer base are hoping to gain a competitive edge but have not tapped into the vast amounts of data they have, especially in solving one of the most acknowledged problems – *customer churn* (i.e., a customer leaving the bank). It is advantageous to banks to know what leads a client to leave the bank. Banks often use the customer churn rate as one of their key business metrics because the cost of retaining existing customers is far less than acquiring new ones, and meanwhile increasing customer retention can greatly increase profits. *Churn prevention* allows companies to develop different programs such as loyalty and retention programs to keep as many customers as possible. Following are the attributes of the dataset we will be working with.
- i. RowNumber (continuous) — corresponds to the record (row) number and has no effect on the output.
 - ii. CustomerId (categorical)— contains random values and has no effect on customer leaving the bank.
 - iii. Surname (categorical)— the surname of a customer has no impact on their decision to leave the bank
 - iv. CreditScore (continuous) — can influence customer churn, since a customer with a higher credit score is less likely to leave the bank.
 - v. Geography (categorical) — a customer’s location can affect their decision to leave the bank.
 - vi. Gender (categorical) — it’s interesting to explore whether gender plays a role in a customer leaving the bank.
 - vii. Age (continuous) — this is certainly relevant, since older customers are less likely to leave their bank than younger ones.
 - viii. Tenure (continuous) — refers to the number of years that the customer has been a client of the bank. Normally, older clients are more loyal and less likely to leave a bank.
 - ix. Balance (continuous) — also a very good indicator of customer churn, as people with a higher balance in their accounts are less likely to leave the bank compared to those with lower balances.
 - x. NumOfProducts (continuous) — refers to the number of products that a customer has purchased through the bank.
 - xi. HasCrCard (categorical) — denotes whether a customer has a credit card. This column is also relevant since people with a credit card are less likely to leave the bank.
 - xii. IsActiveMember (categorical) — active customers are less likely to leave the bank.
 - xiii. EstimatedSalary (continuous) — as with balance, people with lower salaries are more likely to leave the bank compared to those with higher salaries.
 - xiv. Exited (Categorical) — whether or not the customer left the bank. (**Target variable**)
- b. **Data Loading / Preprocessing (10 points)**
- i. Loading (2 points)
 1. Load the data <BankChurn.csv> as a pandas dataframe using the `pd.read_csv()` function which returns a dataframe , store this value in a variable named ‘df’.
 2. The resulting dataframe should have the shape (10000,14) indicating that there are 10000 instances and 14 columns.

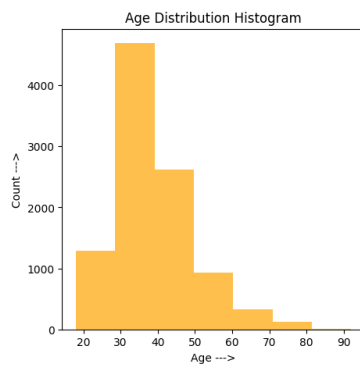
3. In this dataframe, currently you have 9 features which are the following: RowNumber, CustomerID, Surname, CreditScore, Geography, Gender, Age, Tenure, Balance, NumOfProducts, HasCrCard, IsActiveMember, EstimatedSalary. Using the `'pandas.dataframe.drop'` function to drop the RowNumber, CustomerID and Surname columns.
4. Using the `'pandas.isnull()'` function check if there are any missing values in the dataframe and report this value (i.e., the number of missing values per column of the dataframe).

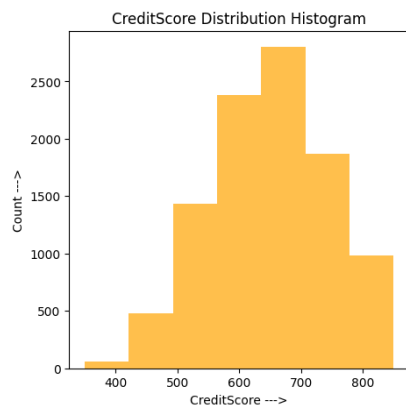
```
CreditScore    0
Geography      0
Gender         0
Age            0
Tenure         0
Balance        0
NumOfProducts  0
HasCrCard      0
IsActiveMember 0
EstimatedSalary 0
Exited         0
dtype: int64
```

5. Your task is to use feature columns to predict the target column (which is categorical in our case). This can be cast as a classification problem.
6. Create a dataframe X of features (by dropping the 'Exited' column from the original dataframe). Create a Pandas Series object of targets Y (by only considering the 'Exited' column from the original dataframe). Moving forward, we will be working with X and Y.

ii. Data Visualization (4 points)

1. Visualize the distribution of the 'Age' and 'CreditScore' column using the `'matplotlib.pyplot.hist'` function as two separate plots. Label the x-axis and the y-axis along with giving the plot a title and assign a bin size of 7.





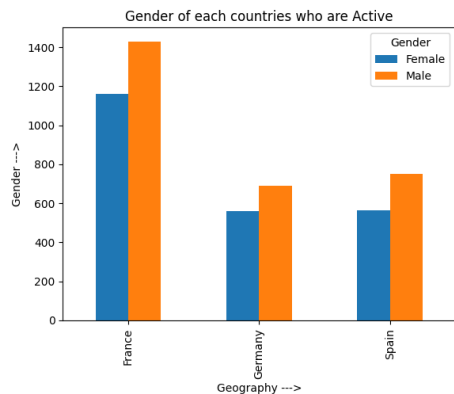
- a. What are the respective mean values of these two features (use the `pandas.DataFrame.mean()` function)?

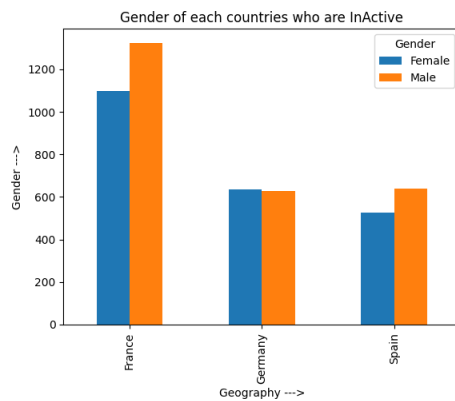
Ans a: The mean value of Age is 38.92 years and CreditScore is \$650.53.

- b. What is the respective standard deviation of these two features (use the `pandas.DataFrame.std()` function)?

Ans b: The Standard deviation of Age is 10.49 years and Creditscore is \$96.65.

2. **Only for this question** use the dataframe consisting of the target variable (initialized as 'df'). Using matplotlib visualize the number of males and females in each country who are active members and not active members. (Visualize this using a **barchart**. You will need to use the 'Gender', 'Geography' and 'IsActiveMember' features for this question). Visualize these graphs on two separate plots with respect to their active status. To create a barchart using matplotlib use the `'matplotlib.pyplot.bar()'` function. Also label the x-axis, y-axis and give the plots a title.





a. How many males are from France and are active members?

Ans a: 1429 males are from France and are active members.

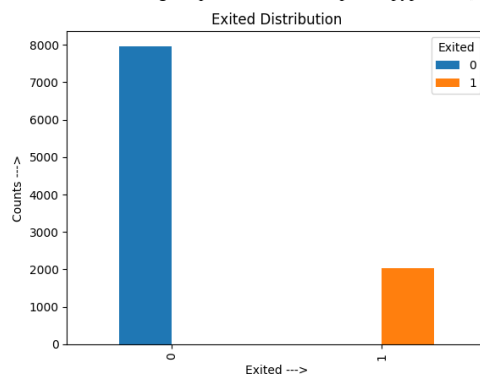
b. How many females are from Spain and are active members?

Ans b: 563 females are from Spain and are active members.

c. How many males are from France or Germany who are not active members?

Ans c: 1951 males are from France or Germany who are not active members.

3. Using the target variable in Y plot a bar chart showing the distribution of the 'Exited' column (To create a barchart using matplotlib use the '`matplotlib.pyplot.bar()`' function).



a. What can be said about this distribution (specifically keeping in mind this distribution represents the target variable) will this have an impact on the results of the classification model?

Ans a: We can say about this distribution will have a more impact on the results of the classification model towards the class 0 of Exited rather than class 1 of Exited.

4. So far you should have successfully been able to load, preprocess and visualize your data. Now, use the `'pd.get_dummies()'` function to convert categorical data into dummy variables ('Gender' and 'Geography'). **(Perform this only on X).**
 - a. What is the shape of X?

Ans a: The shape of X is (10000, 13).

iii. Data Splitting (1 point)

1. Split data into training and test sets using the sklearn `'train_test_split()'` function in a **80:20** ratio. The result of your data split should be `X_train, X_test, y_train, y_test`. (Respectively your training features, testing features, training targets and testing target arrays).

iv. Data Scaling (3 points)

1. Employ the `'MinMaxScaler'` function on the continuous attributes in `X_train`. Employ the `'fit_transform()'` function of the scaler to retrieve the new (scaled) version of the training data (i.e., `fit_transform()` should be run on `'X_train'`). Store the result in `X_train` again.
2. Scale the `X_test` data using the scaler you have just *fit*, this time using the `'transform()'` function. Note: store the scaled values back into `X_test`. At the end of this step, you must have `X_train`, `X_test`, scaled according to the `MinMaxScaler`.

c. Modelling (10 points)

i. (2 points) Modeling (Model Instantiation / Training) using Logistic Regression classifier

1. Employ the `'Logistic Regression classifier'` from sklearn and instantiate the model. Label this model as `'model_1_lr'`
2. Once instantiated, `'fit()'` the model using the *scaled* `X_train, y_train` data.
3. Employ the `'predict()'` function to obtain predictions on `X_test` and store this in a variable labeled as `'y_pred_lr'`.
4. Employ the `'accuracy_score()'` function by using the `'y_pred_lr'` and `'y_test'` variables as the functions parameters and print the accuracy of the Logistic Regression model

ii. (2 points) Modeling (Model Instantiation / Training) using Support Vector Machine Classifier

1. Employ the Support Vector Machine (SVM) classifier from sklearn and instantiate the model. Label this model as `'model_2_svm'`
2. Once instantiated, `'fit()'` the model using the *scaled* `X_train, y_train` data.
3. Employ the `'predict()'` function to obtain predictions on `X_test` and store this in a variable labeled as `'y_pred_svm'`.
4. Employ the `'accuracy_score'` function (`'sklearn.metrics.accuracy()'` function) by using the `'y_pred_lr'` and `'y_test'` variables as the functions parameters and print the accuracy of the SVM model.

iii. (2 points) Modeling Logistic Regression Classifier on a *balanced* dataset

1. Employ Synthetic Minority Oversampling on `X_train` and `y_train`. To use SMOTE you will have to install the imbalanced-learn library, this can either be down by executing the following command `'pip install -U imbalanced-learn'` command `'conda install -c conda-forge imbalanced-learn'` command for the Anaconda Cloud platform. (For more information click the following link: <https://imbalanced-learn.org/stable/install.html>). Import the `'SMOTE'` function from the `'imblearn.over_sampling'`. Use the `'smote.refit_resample()'` function on `X_train` and `y_train` using its default parameters. Store them in `X_train_smote, y_train_smote`. - Be careful to employ SMOTE ONLY on the training data and not on the full dataset because that can cause inadvertent "data leakage" (please see: <https://arxiv.org/pdf/2107.00079.pdf> for details) .

2. Employ a new Logistic Regression classifier from *sklearn* and instantiate the model. Label this model as 'model_3_smote_lr'.
3. Once instantiated, 'fit()' the model using the *balanced* X_train_smote, y_train_smote data.
4. Employ the 'predict()' function to obtain predictions on X_test and store this in a variable labeled as 'y_pred_smote_lr'.
5. Employ the 'accuracy_score' function by using the 'y_pred_lr' and 'y_test' variables as the functions parameters and print the accuracy of the new Logistic Regression model.
 - a. What is your initial observation of the accuracy of model_3_smote_lr vs. accuracy of model_1_lr? What could be the reasoning for (any possible) change in accuracy?

Ans a: The initial observation of the accuracy of model_3_smote_lr vs accuracy of model_1_lr is that accuracy of first model_1_lr was 0.8125 but now the accuracy of third model_3_smote_lr is 0.725. Here we can clearly see the decrease in change of accuracy. The reason for the change in accuracy because it may had made the data more imbalanced which is why it the model is getting low accuracy score.

iv. (2 points) **Modeling SVM on a balanced dataset**

1. Employ Synthetic Minority Oversampling on X_train and y_train. Import the 'SMOTE' function from the 'imblearn.over_sampling'. Use the 'smote.refit_resample()' function on X_train and y_train. Store them in X_train_smote, y_train_smote.
 - a. At the end of this step, your new training set i.e., (X_train_smote , y_train_smote) should have the same number of instances for each of the two classes.
2. Employ a new SVM classifier from *sklearn* and instantiate the model. Label this model as 'model_4_smote_svm'.
3. Once instantiated, 'fit()' the model using the *balanced* X_train_smote, y_train_smote data.
4. Employ the 'predict()' function to obtain predictions on X_test and store this in a variable labeled as 'y_pred_smote_svm'.
5. Employ the 'accuracy_score' function ('sklearn.metrics.accuracy()' function) by using the 'y_pred_lr' and 'y_test' variables as the functions parameters and print the accuracy of the new SVM model.
 - a. What is your initial observation of the accuracy of model_4_smote_svm vs. accuracy of model_2_svm? What could be the reasoning for (any possible) change in accuracy?

Ans a: The initial observation of the accuracy of model_4_smote_svm is lower than model_2_svm accuracy. The reason for the change in accuracy because it may had made the data more imbalanced which is why it the model is getting low accuracy score.

v. (2 points) **Modeling Grid Search Parameter Selection for SVM**

1. We will now be reverting to our X_train and y_train data. Initialize a variable labeled as 'param_grid' storing the following: {'gamma': [0.001, 0.01, 0.1], "C": [1,10,100,1000,10000]}.
2. Employ the gridsearchCV function and initialize the following parameters: estimator = SVC(), param_grid = param_grid, cv=5, verbose=1, scoring = 'accuracy'

3. Once instantiated, 'fit()' the model using the X_train_smote, y_train_smote data.
4. Print the best parameters using the 'best_params_' attribute and print the mean cross validated score of the best estimator (hint use the 'best_score_' attribute).
5. Employ the 'score' function by using the 'X_test' and 'y_test' variables as the functions parameters and print the accuracy of the new gridsearch SVM model.SVM model.

d. **Evaluation (5 points)**

- i. (2 points) Calculate F1 Score, Precision, Recall, Accuracy (All on the test set X_test, y_test)
 1. Employ the 'classification_report()' function from sklearn.metrics to report the precision recall, f1 score and accuracy for each class for the first **four** models (**parts c.i – c.iv**).
- ii. (2 points) Visualize a confusion matrix for the first four models
 1. Employ the 'confusion_matrix()' function from sklearn.metrics to report the confusion matrix results.
 2. Report the False Negative and False Positive values for model_1_lr.

Ans 2: False Negative 318 and False Positive 57 values for model_1_lr

- iii. (1 point) Report the best F1 score of the grid search implemented in the fifth model (**part c.v**). Also report the best parameters from the grid search on the training set.

Ans iii: The best F1 score of the grid seach implemented in the fifth model (part c.v) is 0.5111492281303602

The best parameters from the grid search on the training set is {'C': 10000, 'gamma': 0.01 }

e. **Discussion (8 points)**

- i. (2 points) Compare the performance of the Logistic Regression (model_1_lr) classifier and the SVM classifier (model_2_svm). Using the classification reports which model performs better? Mention the specific numbers (i.e., from your results calculating the precision, recall, F1 score) in a table.

Ans i:

Model Name	Class	Precision	Recall	F1
model_1_lr	0	0.83	0.96	0.89
model_1_lr	1	0.57	0.19	0.29
model_2_svm	0	0.84	0.99	0.91
model_2_svm	1	0.85	0.23	0.36

Using classification reports SVM classifier (model_2_svm) model preforms better than Logistic Regression (model_1_lr) classifier.

- ii. (2 points) Compare the results between the models prior to SMOTE and after. Use a combination of the classification report and the confusion matrix to emphasize your observation. (model_1_lr vs model_3_lr and model_2_svm vs model_4_svm)

Ans ii:

Model Name	Class	Precision	Recall	F1
model_1_smote_lr	0	0.91	0.73	0.81
model_1_smote_lr	1	0.39	0.71	0.50
model_2_smote_svm	0	0.91	0.79	0.85
model_2_smote_svm	1	0.44	0.69	0.54

By comparing the results between the models prior to SMOTE and after we can say that the model_1_lr performs better than model_3_smote_lr and model_2_svm performs better than model_4_smote_svm.

- iii. (2 points) Discuss the advantages and disadvantages between oversampling and under-sampling. Use the following article to help you formulate your answer and reasoning.
(<https://www.mastersindatascience.org/learning/statistics-data-science/undersampling/>)

Ans iii: Oversampling is appropriate when data scientists do not have enough information. Oversampling methods duplicate or create new synthetic examples in the minority class. Oversampling is basically upsampling the minority class to make it almost equal in size with respect to the majority class.

Undersampling is appropriate when there is plenty of data for an accurate analysis. Undersampling methods delete or merge examples in the majority class. Undersampling is basically downsampling the majority class to make it (almost) equal in size with respect to the minority class.

- iv. (2 points) From the results of the grid-search, comment on importance of the gamma parameter on SVM performance. What effect will the gamma value have if it is too big or too small?

Ans iv: The results of the grid-search, the gamma parameter which we get 0.01. The model accuracy lowers with the increasing value of gamma. The lower value of gamma will mean that the data points have very high radius of influence. That's why gamma parameter plays important role for SVM performance. The gamma parameter defines how far the influence of a single training example reaches, with small values meaning 'far' and big values meaning 'close'. Also, big value of gamma will mean that radius of influence is limited to only support vectors. This would essentially mean that the model tries and overfit.