Mini-Project: Stellar Classification

This document briefly explains the idea and observation required to complete the Milestone-2 provided for the dataset, understanding of the learning tasks developed by other data scientist/ analyst or engineers commonly known as developers, how they implemented different type of models to achieve an accuracy of above 95%. Also analyze and understand their code and different techniques they have implemented.

Dataset link: https://www.kaggle.com/fedesoriano/stellar-classification-dataset-sdss17

Learning Tasks developed by others:

There are quite a few developers who have considered this learning tasks for the "Stellar Classification Dataset". But I have chosen to analyze the code developed by top three developers who have received most votes from others and achieved higher accuracy of above 95 %. These developers are BATUCAN SENKAL, BEYZA NUR NAKKAS and WALEED FAHEEM.

After reviewing their code, I can understand their various approaches to deploy different types of classifiers for the Stellar Classification. I am more impressed with the techniques which are adopted by both two developers - BATUCAN SENKAL and BEYZA NUR NAKKAS, who achieved an accuracy of about 98%.

Reference links to the developer's code:

https://www.kaggle.com/code/psycon/stars-galaxies-eda-and-classification

https://www.kaggle.com/code/beyzanks/stellar-classification-98-4-acc-100-auc#Classifiers

https://www.kaggle.com/code/waleedfaheem/stellar-classification-and-supervised-learning

Keys steps implemented by the developers and how they accomplished these tasks:

1. Numeric Correlation: In this code, the developer generates a subplot with two correlation heatmaps (Pearson and Spearman) and then he customizes their appearance and layout. Finally, it provided a visual representation of the relationships between numerical variables in the dataset. This helps us to identify the patterns and correlations.

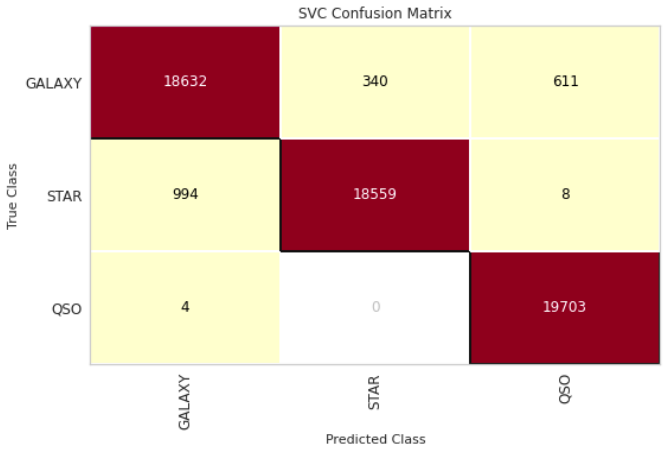
```
fig = make_subplots(rows=2, cols=1,shared_xaxes=True,subplot_titles=('Perason Correaltion',  'Spearm
an Correaltion'))
colorscale=     [[1.0              , "rgb(165,0,38)"],
                [0.8888888888888888, "rgb(215,48,39)"],
                [0.7777777777777778, "rgb(244,109,67)"],
                [0.6666666666666666, "rgb(253,174,97)"],
                [0.5555555555555556, "rgb(254,224,144)"],
                [0.4444444444444444, "rgb(224,243,248)"],
                [0.3333333333333333, "rgb(171,217,233)"],
                [0.2222222222222222, "rgb(116,173,209)"],
                [0.1111111111111111, "rgb(69,117,180)"],
                [0.0               , "rgb(49,54,149)"]]
```

```
s_val =df.corr('pearson')
s_idx = s_val.index
s_col = s_val.columns
s_val = s_val.values
fig.add_trace(
    go.Heatmap(x=s_col,y=s_idx,z=s_val,name='pearson',showscale=False,xgap=0.7,ygap=0.7),
    row=1, col=1
)


s_val =df.corr('spearman')
s_idx = s_val.index
s_col = s_val.columns
s_val = s_val.values
fig.add_trace(
    go.Heatmap(x=s_col,y=s_idx,z=s_val,xgap=0.7,ygap=0.7),
    row=2, col=1
)
```

```
fig.update_layout(
    hoverlabel=dict(
        bgcolor="white",
        font_size=16,
        font_family="Rockwell"
    )
)
fig.update_layout(height=700, width=900, title_text="Numeric Correaltions")
fig.show()
```

2. Confusion Matrix: Developer is performing a high-level analysis about the classifier's performance using a confusion matrix to classify the objects into different classes. Then the developer fit, score, and show the confusion matrix to provide insight about the accuracy and error rates for each of the true class (GALAXY, STAR and QUASAR)

```
svm_cm = ConfusionMatrix(svm_clf, classes=['GALAXY','STAR','QSO'])

svm_cm.fit(x_train, y_train)
svm_cm.score(x_test, y_test)
svm_cm.show()
```



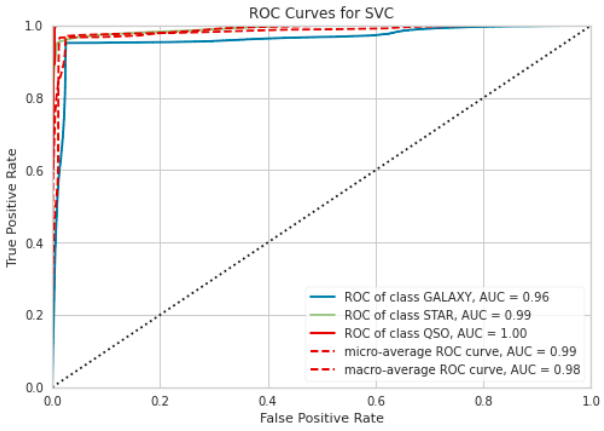
3. Classification Report: Generate a classification report to display a concise overview about how the model performs in terms of precision, recall, f1-score, and the accuracy.

```
print(classification_report(y_test, predicted))
```

	precision	recall	f1-score	support
0	0.95	0.96	0.96	16691
1	0.97	1.00	0.99	16693
2	0.98	0.96	0.97	16805
accuracy			0.97	50189
macro avg	0.97	0.97	0.97	50189
weighted avg	0.97	0.97	0.97	50189

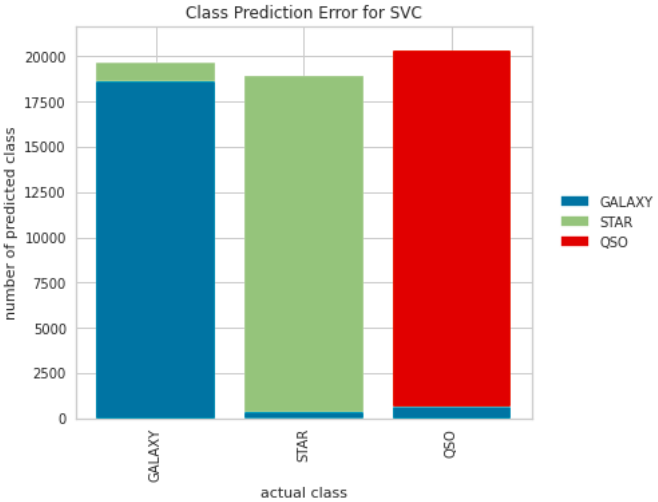
4. ROC Curve: Create a visual representation of the classifier performance by showing the ROC curve and AUC score for true positive and false positive rates. These metrics are generally used by the developers to assess how the classifier distinguishes between different classes (GALAXY, START and QUOSAR)

```
visualizer = ROCAUC(svm_clf, classes=['GALAXY','STAR','QSO'])

visualizer.fit(x_train, y_train)
visualizer.score(x_test, y_test)
visualizer.show()
```



5. Class Prediction Error: Developer is generating a visual representation of a class prediction errors with number of predicted class vs the actual class. 'classes' likely to specifies the classes for which prediction errors should be analyzed. This is a very useful to understand if the model is making any mistakes in its predictions.

```
visualizer = ClassPredictionError(svm_clf, classes=['GALAXY','STAR','QSO'])

visualizer.fit(x_train, y_train)
visualizer.score(x_test, y_test)
visualizer.show()
```



6. Validating different types of classifier models: The code iterates through a series of model's dictionary, to calculate and print the accuracy score on a test dataset (X_test, y_test). This accuracy score will represent how well the model predicts the target variable. The accuracy scores can be used to assess the quality of predictions generated by these models. Also, this code validates the performance of CatBoostClassifier and LGBMClassifier using a cross-validation.

```
#Checking the validy of SVM, DecisionTree and random forest classifier model.....

for model_name, model in models.items():
    print(model_name + " {:.2f}%".format(model.score(X_test, y_test) * 100))
```

```
Linear Support Vector Machine :  90.61%
Decision Tree :  95.24%
Random Forest Classifier :  97.46%
```

```
#Validating CatBoostClassifier model here using cross validation.....

model = CatBoostClassifier(verbose  = 0)
score = cross_val_score(model, X, y, cv = skfold)
print('CatBoostClassifier : ' + " {:.2f}%".format(np.mean(score) * 100))
```

```
CatBoostClassifier :  97.56%
```

```
#Validating LGBC model here using cross validation.....

model = LGBMClassifier()
score = cross_val_score(model , X , y ,cv=skfold)
print('Light GradientBoostingClassifier : ' + " {:.2f}%".format(np.mean(score) * 100))
```

```
Light GradientBoostingClassifier :  97.70%
```