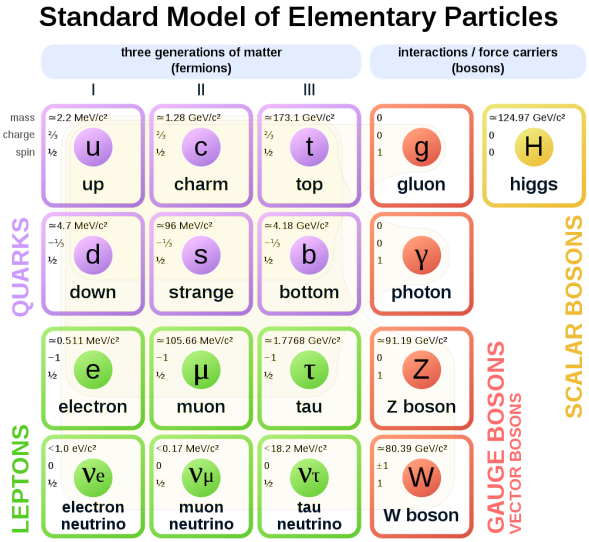
Project Proposal: Flavors of Physics, The Strange D Meson Decay

Domain Background

This project is a particle physics problem. Its name is inspired by what physicists call "flavor", the species of an elementary particle. The Standard Model of particle physics is a well-established theory that explains the properties of fundamental particles and their interactions, describing the "flavor" of each particle. As mentioned in Charlotte Louise Mary Wallace CERN Thesis, the Standard Model theory has been tested by multiple experiments, but despite its successes, it is still incomplete, and further research is needed.

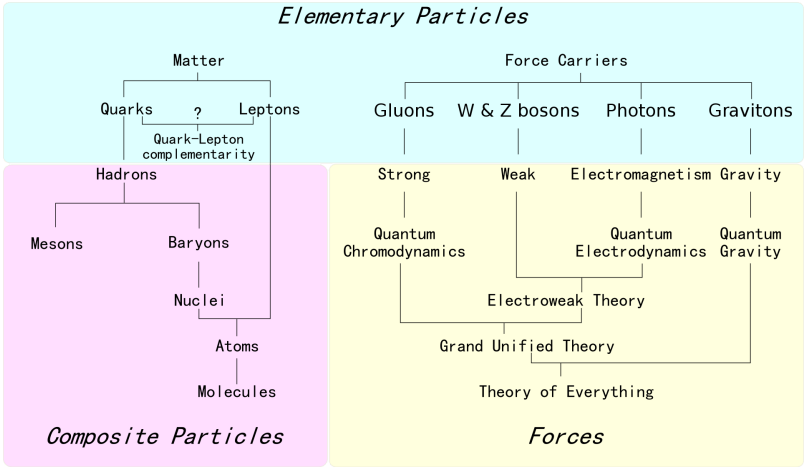
The Standard Model counts six flavors of quarks and six flavors of leptons, as shown below. "Flavor" is essentially a quantum number that characterizes the quantum state of those quarks.



The Ds decay project is influenced by a CERN kaggle competition problem about the flavors of physics. In the initial problem, scientists try to find if it is possible the τ (*tau*) lepton to decay (transform into multiple other particles) to three μ (muon) leptons. The problem I chose, however, concerns the Ds meson or *strange D meson*, a composite particle that consists of one quark or one antiquark, and how often it decays into a ϕ (phi meson) and a π (pi meson or pion) based on multiple factors. The decay is described by the following flow:

$$D_s \rightarrow \varphi\pi$$

You can see where the meson belongs in the subatomic particles map below. The purple part describes the composite particles.



Ander Ryd in his paper argues that the D meson decays have been a challenge, though scientists have been focused on their decays since the particle discovery. As a result, the existing dataset of this project is sufficient and based on well-studied experiment observations.

Problem Statement

The problem falls into the category of binary classification problems. Based on particle collision events (that cause the $D_s \rightarrow \varphi\pi$ decays) and their properties, I am challenged to predict whether the decay we are interested in happens in a collision or not.

Datasets and Inputs

As described in the flavors of physics project, the $D_s \rightarrow \varphi\pi$ decay has a very similar topology as the *tau* decay and their datasets share almost the same features. In the *tau* decay problem, the Ds decay data is used as part of the CERN evaluation process. This dataset will be used as the main dataset of the $D_s \rightarrow \varphi\pi$ decay problem solution.

This is a labeled dataset of 16.410 samples and 46 features, which are described below. The *signal* column classifies the samples into *signal events* (decays happening) denoted with *1* and *background events* (decays not happening) denoted with *0*. The dataset is balanced, with 8.205 signal events and 8.205 background events.

The features of the dataset are described below:

- FlightDistance - Distance between Ds and PV (primary vertex, the original protons collision point).
- FlightDistanceError - Error on FlightDistance.
- LifeTime - Life time of Ds candidate.
- IP - Impact Parameter of Ds candidate.
- IPSig - Significance of Impact Parameter.
- VertexChi2 - χ2 of Ds vertex.
- dira - Cosine of the angle between the Ds momentum and line between PV and *Ds* vertex.
- pt - transverse momentum of Ds.
- DOCAone - Distance of Closest Approach between p0 and p1.
- DOCAtwo - Distance of Closest Approach between p1 and p2.
- DOCAthree - Distance of Closest Approach between p0 and p2.
- IP_p0p2 - Impact parameter of the p0 and p2 pair.
- IP_p1p2 - Impact parameter of the p1 and p2 pair.

- isolationa - Track isolation variable.
- isolationb - Track isolation variable.
- isolationc - Track isolation variable.
- isolationd - Track isolation variable.
- isolatione - Track isolation variable.
- isolationf - Track isolation variable.
- iso - Track isolation variable.
- CDF1 - Cone isolation variable.
- CDF2 - Cone isolation variable.
- CDF3 - Cone isolation variable.
- ISO_SumBDT - Track isolation variable.
- p0_IsoBDT - Track isolation variable.
- p1_IsoBDT - Track isolation variable.
- p2_IsoBDT - Track isolation variable.
- p0_track_Chi2Dof - Quality of p0 muon track.
- p1_track_Chi2Dof - Quality of p1 muon track.
- p2_track_Chi2Dof - Quality of p2 muon track.
- p0_pt - Transverse momentum of p0 muon.
- p0_p - Momentum of p0 muon.
- p0_eta - Pseudorapidity of p0 muon.
- p0_IP - Impact parameter of p0 muon.
- p0_IPSig - Impact Parameter Significance of p0 muon.
- p1_pt - Transverse momentum of p1 muon.
- p1_p - Momentum of p1 muon.
- p1_eta - Pseudorapidity of p1 muon.
- p1_IP - Impact parameter of p1 muon.
- p1_IPSig - Impact Parameter Significance of p1 muon.
- p2_pt - Transverse momentum of p2 muon.
- p2_p - Momentum of p2 muon.
- p2_eta - Pseudorapidity of p2 muon.
- p2_IP - Impact parameter of p2 muon.
- p2_IPSig - Impact Parameter Significance of p2 muon.
- SPDhits - Number of hits in the SPD detector.
- signal - This is the target variable.

Obtain the dataset

There are three ways to get the data described above:

- I recommend downloading the resampled dataset from the Github repo I created for this project. I intend to use this resampled dataset, as the original is heavily imbalanced. The resampled dataset is also smaller and easier to manage, which makes it more suitable for this Udacity project. I made sure that the dataset has sufficient data for my analysis. If you want to get the original dataset, follow the next point.
- From Kaggle, by downloading the *check_agreement.csv.zip* from here (this requires a Kaggle account).

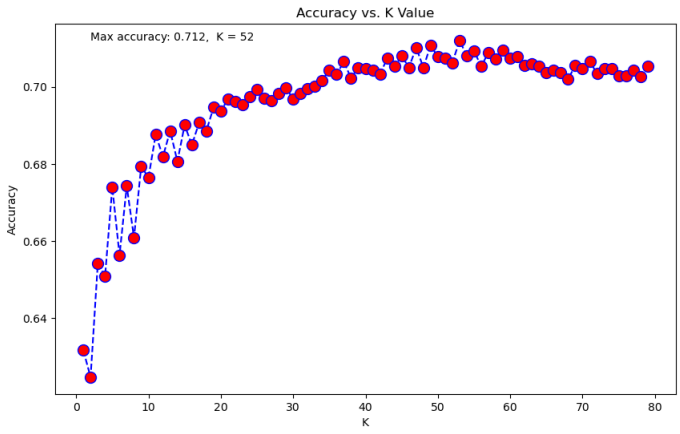
> Note that in the resampled dataset, I have dropped the "weights" feature from the original dataset as, according to the description of the dataset, is the feature used to determine if the decay happens or not based on its value, and is the one used to create the binary *signal* column. It will not be used in the solution whatsoever.

Solution Statement

This is a binary classification problem, so the solution will be the output of a binary classifier. There is no constrain in using any classifier in particular for this problem. However, in the evaluation description of the Kaggle competition described so far, is it mentioned that the Kolmogorov–Smirnov (KS) test is used to evaluate the differences between the classifier distribution and the true *signal* values distribution. I intent to train multiple models in combination with different data cleaning methods, but in the final solution, I will only present the chosen data cleaning method and model based on evaluation from different performance metrics, including the KS test.

Benchmark Model

As a benchmark model, I use a simple k-Nearest Neighbor classifier, trained with the resampled data, and grid search for tuning the k hyperparameter. The benchmark model script can be found here. The execution of that script generates the following plot. The plot shows the accuracy of the kNN model for each one of the k values (from 1 to 80). The best model is the kNN model with k=52, and highest accuracy of 71%.



Evaluation Metrics

The evaluation metric used to choose the k values of the kNN benchmark model is the `sklearn` accuracy. It is calculated by dividing the number of correct predictions by the total number of samples. As the dataset is balanced with equal class distribution, the accuracy paradox is avoided and the metric does not provide missleading information about the models performance.

Project Design

A high-level workflow for the solution approach is the following:

1. **Understand the problem - identify the problem category**: I already know that this is a binary classification problem, which will help me choose the models to train.

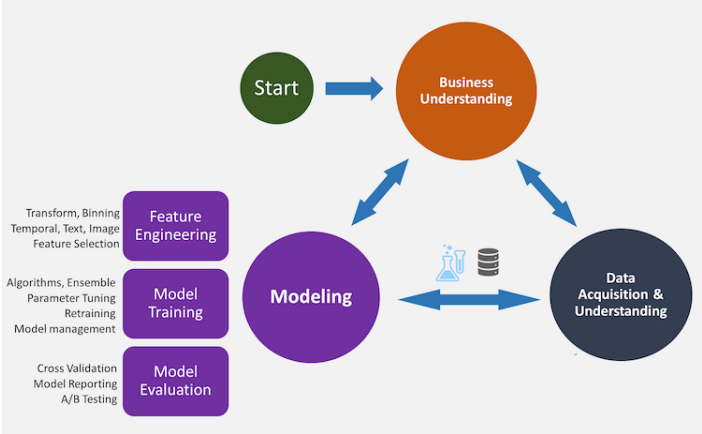
2. **Data mining**: Obtain the necessary datasets for this analysis, as mentioned in the associated section of data input above.

3. **Data Visualization, Cleaning, and Engineering**: Check if the data is balanced, if there is a linear correlation between the features to determine which features to keep, or check if the data will work better with a specific scaling. Distribution plots, or even dimensionality reduction techniques, may help with the data visualization.

4. **Model Training/Tuning**: Train a selection of binary classification algorithms, trying different data scales, different number of features, and cross-validation to avoid overfitting. Make use of search techniques like grid or random search to tune the hyperparameters. Models like Support Vector Machines, Stochastic Gradient Descent, XGBoost, CatBoost, LightGBM or even a custom ensemble classifier might be suitable for this problem.

5. **Model Evaluation**: A number of performance metrics will be used for this step. Evaluation metrics suitable for this problem might be false positive rate, false negative rate, true negative rate, negative predictive value, accuracy or Kolmogorov–Smirnov test.

A workflow that sums up the steps above is showm below. This workflow is part of a typical data science lifecycle, as presented here.



Why I chose this project?

I have a bachelor's degree in physics, but I have not worked as a physicist so far. I started having an interest in data science in recent years, and I am now confident enough to start utilizing my data science knowledge in solving physics problems, combining my passion for both fields. I specifically chose a CERN particle physics project as I know that scientists at CERN often provide data to the public based on their observations. I was able to find both an interesting machine learning problem and a dataset online to work with, with sufficient data for a data science analysis. I initially started working on the tau decay problem, but even simple classifiers predict the signal with high performance without much hyperparameters tuning due to the nature of the dataset. This problem would not help me learn as much as a problem that requires further model improvements to get high-performing results, so I switched to the Ds decay which is a similar problem with lower benchmark performance.

Sources:

- https://en.wikipedia.org/wiki/Flavour_(particle_physics)
- https://cds.cern.ch/record/2713513?ln=en

- https://en.wikipedia.org/wiki/Standard_Model
- https://cds.cern.ch/record/2196092/files/CERN-THESIS-2016-064.pdf
- https://en.wikipedia.org/wiki/Particle_decay
- https://en.wikipedia.org/wiki/D_meson
- https://en.wikipedia.org/wiki/Phi_meson
- https://en.wikipedia.org/wiki/Pion
- https://wiki.classe.cornell.edu/pub/People/AndersRyd/DHadRMP.pdf
- https://www.kaggle.com/c/flavours-of-physics/overview
- https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test
- https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/lifecycle
- http://cds.cern.ch/record/2668282/files/BPH-17-004-pas.pdf
- https://neptune.ai/blog/evaluation-metrics-binary-classification
- https://machinelearningmastery.com/failure-of-accuracy-for-imbalanced-class-distributions/
- https://en.wikipedia.org/wiki/Accuracy_paradox
- http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice/
- https://www.kaggle.com/arthurtok/introduction-to-ensembling-stacking-in-python