Module 1 - Fundamentals of AI

Course Number: EAI 6000

Academic Term: Fall 2020 CPS Analytics

Instructor's Name: Kasun Samarasinghe

Assignment Completion Date: 11-01-2020

Submitted By:

Dhiren Vasudev Pagrani

Sunil Raj Thota

Shivani Sharma

INTRODUCTION

This is our group assignment and we have selected the data set from Kaggle which is based on the medical research conducted by MIT and Harvard which is serving the purpose of developing a drug advancement through enhancements to “Mechanisms of Action (MoA)” prediction algorithms. The goal here is to improvise the algorithm already implemented by MIT and Harvard which provides the classification of drugs based on the biological activities happening as a result of medicines with the body cells.

In pharmacology, the term “Mechanism of Action(MOA)” alludes to the particular biochemical association through which a medication substance creates its pharmacological impact. A component of activity for the most part incorporates the notice of the particular atomic focuses to which the medication ties, for example, a catalyst or receptor. Receptor locales have explicit affinities for drugs dependent on the synthetic structure of the medication, just as the particular activity that happens there.

One methodology is to treat an example of human cells with the medication and afterward investigate the cell reactions with calculations that look for likeness to known examples in huge genomic information bases, for example, libraries of quality articulation or cell feasibility examples of medications with known MoAs.

In this project, we will approach a novel dataset that consolidates quality articulation and cell practicality information. In light of the MoA explanations, the precision of arrangements will be assessed on the normal estimation of the logarithmic misfortune work applied to each medication MoA comment pair.

Now we would like to explain the EDA part done in the analysis section.

ANALYSIS

So, we have first imported the libraries for the analysis of the data and reading the data files, and storing them in a data frame:

In [1]:
```
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [17]:
```
# reading data files and storing them in a dataframe

df_train_features = pd.read_csv('D:/Fundamental of AI/lish-moa/train_features.csv')
df_test_features = pd.read_csv('D:/Fundamental of AI/lish-moa/test_features.csv')
df_train_target_nonscored = pd.read_csv('D:/Fundamental of AI/lish-moa/train_targets_nonscored.csv')
df_train_target_scored = pd.read_csv('D:/Fundamental of AI/lish-moa/train_targets_scored.csv')
```

Checking the dataset for the presence of null values:

In [18]:
```
# testing for the null values in all the datasets
df_train_features.isnull().sum()
df_test_features.isnull().sum()
df_train_target_nonscored.isnull().sum()
df_train_target_scored.isnull().sum()
```

Out[18]:
```
sig_id                                0
5-alpha_reductase_inhibitor           0
11-beta-hsd1_inhibitor                0
acat_inhibitor                        0
acetylcholine_receptor_agonist        0
                                     ..
ubiquitin_specific_protease_inhibitor 0
vegfr_inhibitor                       0
vitamin_b                             0
vitamin_d_receptor_agonist            0
wnt_inhibitor                         0
Length: 207, dtype: int64
```

Counting the number of rows and columns in the dataset before starting the analysis part:

In [19]:
```
# couting number of rows and columns in training datasets
print('count of rows: ', df_train_features.shape[0])
print('count of cols: ', df_train_features.shape[1])

df_train_features.head()
```

```
count of rows:  23814
count of cols:  876
```

Out[19]:

	sig_id	cp_type	cp_time	cp_dose	g-0	g-1	g-2	g-3	g-4	g-5	...	c-90	c-91	c-92	c-93	c-94	c-95	
0	id_000644bb2	trt_cp	24	D1	1.0620	0.5577	-0.2479	-0.6208	-0.1944	-1.0120	...	0.2862	0.2584	0.8076	0.5523	-0.1912	0.6584	-0
1	id_000779bfc	trt_cp	72	D1	0.0743	0.4087	0.2991	0.0604	1.0190	0.5207	...	-0.4265	0.7543	0.4708	0.0230	0.2957	0.4899	0
2	id_000a6266a	trt_cp	48	D1	0.6280	0.5817	1.5540	-0.0764	-0.0323	1.2390	...	-0.7250	-0.6297	0.6103	0.0223	-1.3240	-0.3174	-0
3	id_0015fd391	trt_cp	48	D1	-0.5138	-0.2491	-0.2656	0.5288	4.0620	-0.8095	...	-2.0990	-0.6441	-5.6300	-1.3780	-0.8632	-1.2880	-1
4	id_001626bd3	trt_cp	72	D2	-0.3254	-0.4009	0.9700	0.6919	1.4180	-0.8244	...	0.0042	0.0048	0.6670	1.0690	0.5523	-0.3031	0

5 rows × 876 columns

In [20]:
```
# couting number of rows in non scored dataset
print('count of rows: ', df_train_target_nonscored.shape[0])
print('count of cols: ', df_train_target_nonscored.shape[1])

df_train_target_nonscored.head()
```

```
count of rows:  23814
count of cols:  403
```

Out[20]:

	sig_id	abc_transporter_expression_enhancer	abl_inhibitor	ace_inhibitor	acetylcholine_release_enhancer	adenosine_deaminase_inhibitor	adenosir
0	id_000644bb2	0	0	0	0	0	
1	id_000779bfc	0	0	0	0	0	
2	id_000a6266a	0	0	0	0	0	
3	id_0015fd391	0	0	0	0	0	
4	id_001626bd3	0	0	0	0	0	

5 rows × 403 columns

In [21]:
```
# couting number of rows in scored dataset
print('count of rows: ', df_train_target_scored.shape[0])
print('count of cols: ', df_train_target_scored.shape[1])

df_train_target_scored.head()
```

```
count of rows:  23814
count of cols:  207
```

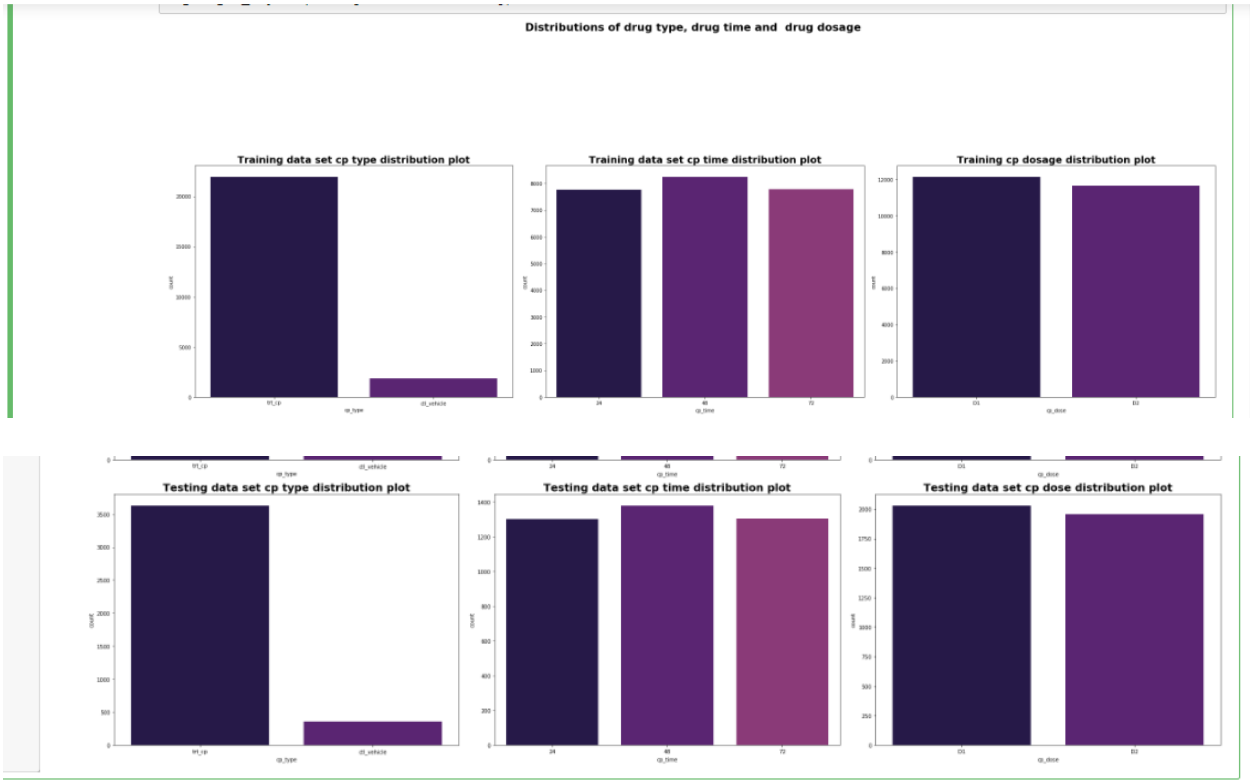
Out[21]:

	sig_id	5-alpha_reductase_inhibitor	11-beta-hsd1_inhibitor	acat_inhibitor	acetylcholine_receptor_agonist	acetylcholine_receptor_antagonist	acetylcholineste
0	id_000644bb2	0	0	0	0	0	
1	id_000779bfc	0	0	0	0	0	
2	id_000a6266a	0	0	0	0	0	
3	id_0015fd391	0	0	0	0	0	
4	id_001626bd3	0	0	0	0	0	

5 rows × 207 columns

Now we have analyzed with the help of visualization we created to understand the pattern and relationship between drug type, duration, and dosage of drugs.

In [26]:
```
# analysis on the drug type, drug effect duration and dosage of the drug in both training and testing data sets.
fig, ax = plt.subplots(2, 3, figsize=(30, 20))
sns.set_palette(sns.color_palette("magma"))

sns.countplot(df_train_features['cp_type'], ax=ax[0][0])
ax[0][0].set_title('Training data set cp type distribution plot', fontsize=20, fontweight='bold')

sns.countplot(df_train_features['cp_time'], ax=ax[0][1])
ax[0][1].set_title('Training data set cp time distribution plot', fontsize=20, fontweight='bold')

sns.countplot(df_train_features['cp_dose'], ax=ax[0][2])
ax[0][2].set_title('Training cp dosage distribution plot', fontsize=20, fontweight='bold')

sns.countplot(df_test_features['cp_type'], ax=ax[1][0])
ax[1][0].set_title('Testing data set cp type distribution plot', fontsize=20, fontweight='bold')

sns.countplot(df_test_features['cp_time'], ax=ax[1][1])
ax[1][1].set_title('Testing data set cp time distribution plot', fontsize=20, fontweight='bold')

sns.countplot(df_test_features['cp_dose'], ax=ax[1][2])
ax[1][2].set_title('Testing data set cp dose distribution plot', fontsize=20, fontweight='bold')

fig.suptitle('Distributions of drug type, drug time and  drug dosage', fontsize=22, fontweight='bold')

fig.tight_layout(rect=[0, 0.05, 1, 0.80]);
```



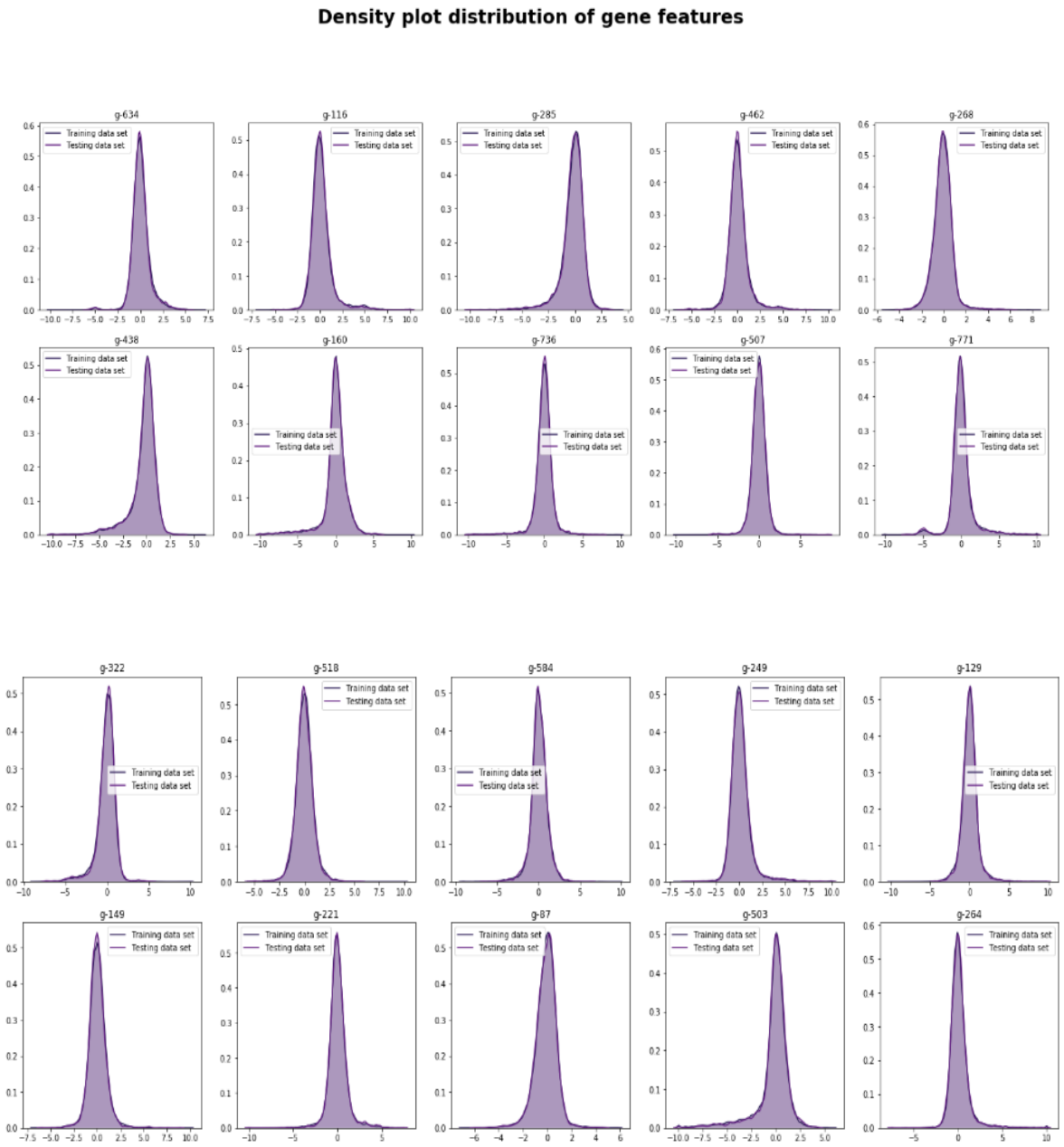
Now we decided to deep dive into the gene and cell data factors provided in the data set and here is the analysis we did for that:

In [31]:
```
# calculation of count of gene features in the training data set

gene_features_count_calculation = list([x for x in list(df_train_features.columns) if "g-" in x])

print(len(gene_features_count_calculation))
```

```
772
```

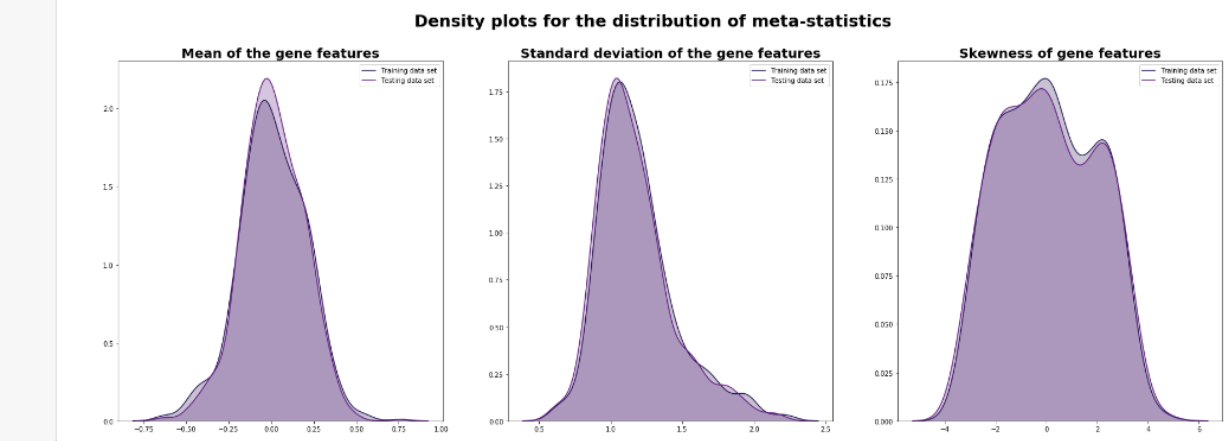
In [37]:

```
# Calculation of meta data stats for the features for making a plot for the distribution to show features
fig, ax = plt.subplots(4, 5, figsize=(25, 20))
rand_feats = np.random.choice(gene_features_count_calculation, 20, replace=False)

fig.suptitle(' Density plot distribution of gene features', fontsize=25, fontweight="bold")

for x in range(20):
    i = x // 5
    j = x % 5

    sns.kdeplot(df_train_features[rand_feats[x]], shade=True, label="Training data set", ax=ax[i][j])
    sns.kdeplot(df_test_features[rand_feats[x]], shade=True, label="Testing data set", ax=ax[i][j])
    ax[i][j].set_title(rand_feats[x])
```



In [46]:
```
# calculation and plotting of meta statistics of training and testing data sets  i.e mean, standard deviation  and skewness

fig, ax = plt.subplots(1, 3, figsize=(30, 10))
fig.suptitle('Density plots for the distribution of meta-statistics', fontsize=25, fontweight="bold")

sns.kdeplot(df_train_features[gene_features_count_calculation].mean(), shade=True, label="Training data set", ax=ax[0])
sns.kdeplot(df_test_features[gene_features_count_calculation].mean(), shade=True, label="Testing data set", ax=ax[0])
ax[0].set_title("Mean of the gene features", fontsize=20, fontweight="bold")

sns.kdeplot(df_train_features[gene_features_count_calculation].std(), shade=True, label="Training data set", ax=ax[1])
sns.kdeplot(df_test_features[gene_features_count_calculation].std(), shade=True, label="Testing data set", ax=ax[1])
ax[1].set_title("Standard deviation of the gene features", fontsize=20, fontweight="bold")

sns.kdeplot(df_train_features[gene_features_count_calculation].skew(), shade=True, label="Training data set", ax=ax[2])
sns.kdeplot(df_test_features[gene_features_count_calculation].skew(), shade=True, label="Testing data set", ax=ax[2])
ax[2].set_title("Skewness of gene features", fontsize=20, fontweight="bold")
```

Out[46]: Text(0.5,1,'Skewness of gene features')



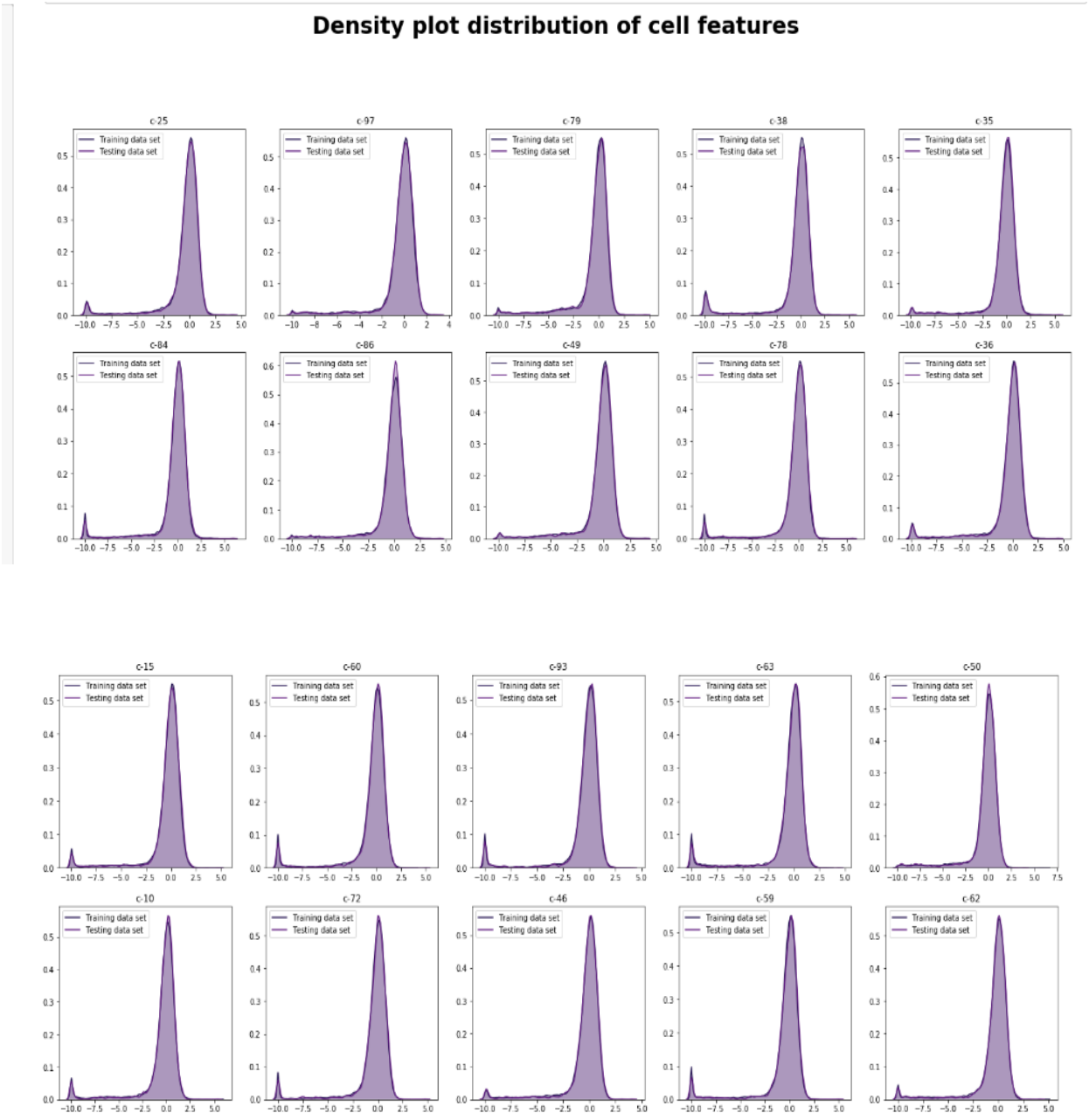
From the above plots, we can infer that mostly feature distributions are centered at 0 and mostly are symmetric, and very few are right or left-skewed.

Now following is the analysis we did for the data provided for the gene and cell features collection for each test conducted.

In [40]:
```
# analysis on the cell features
# calculation of count of gene features in the training data set

cell_feature_calculation = list([x for x in list(df_train_features.columns) if "c-" in x])

print(len(cell_feature_calculation))
```

```
100
```

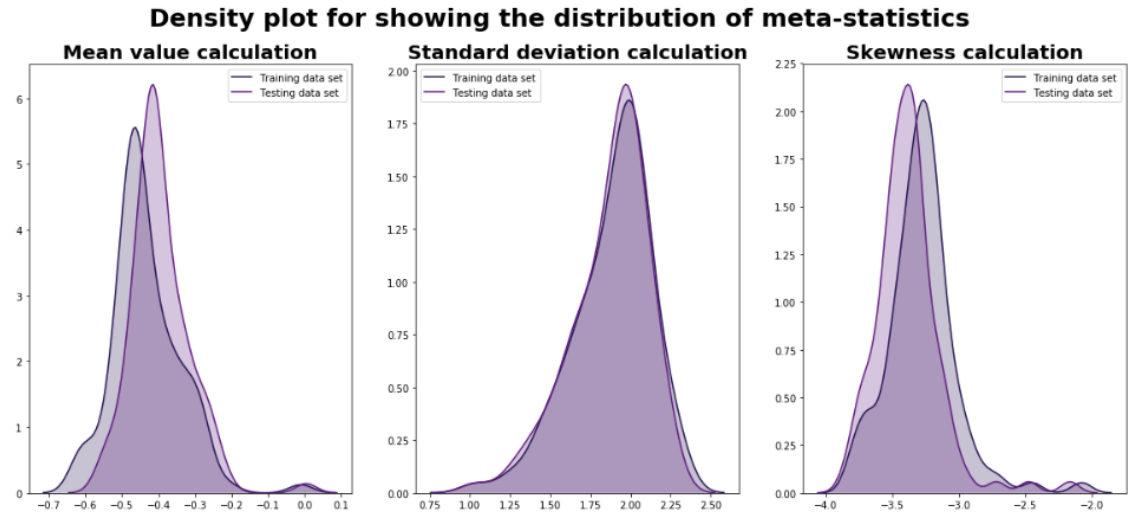
In [43]:
```
# Calculation of meta data stats for the cell features for making a plot for the distribution to show features
fig, ax = plt.subplots(4, 5, figsize=(25, 20))
rand_feats = np.random.choice(cell_feature_calculation, 20, replace=False)

fig.suptitle('Density plot distribution of cell features', fontsize=30, fontweight="bold")

for x in range(20):
    i = x // 5
    j = x % 5
    sns.kdeplot(df_train_features[rand_feats[x]], shade=True, label="Training data set", ax=ax[i][j])
    sns.kdeplot(df_test_features[rand_feats[x]], shade=True, label="Testing data set", ax=ax[i][j])
    ax[i][j].set_title(rand_feats[x])
```



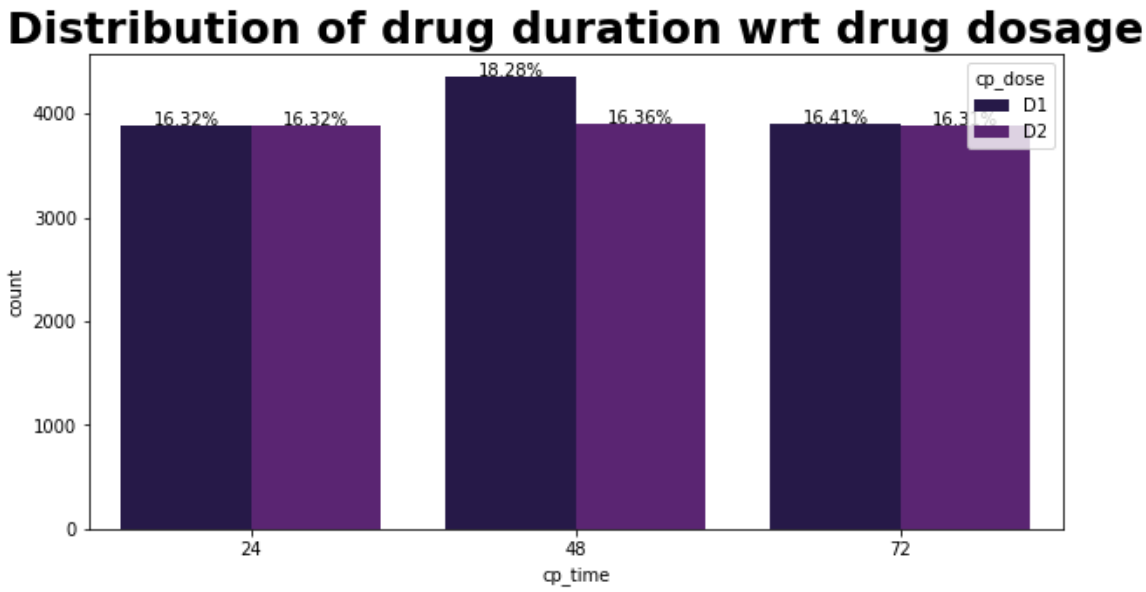
In [47]:
```
fig, ax = plt.subplots(1, 3, figsize=(20, 8))
fig.suptitle('Density plot for showing the distribution of meta-statistics', fontsize=25, fontweight="bold")

sns.kdeplot(df_train_features[cell_feature_calculation].mean(), shade=True, label="Training data set", ax=ax[0])
sns.kdeplot(df_test_features[cell_feature_calculation].mean(), shade=True, label="Testing data set", ax=ax[0])
ax[0].set_title("Mean value calculation", fontsize=20, fontweight="bold")
sns.kdeplot(df_train_features[cell_feature_calculation].std(), shade=True, label="Training data set", ax=ax[1])
sns.kdeplot(df_test_features[cell_feature_calculation].std(), shade=True, label="Testing data set", ax=ax[1])
ax[1].set_title("Standard deviation calculation", fontsize=20, fontweight="bold")

sns.kdeplot(df_train_features[cell_feature_calculation].skew(), shade=True, label="Training data set", ax=ax[2])
sns.kdeplot(df_test_features[cell_feature_calculation].skew(), shade=True, label="Testing data set", ax=ax[2])
ax[2].set_title("Skewness calculation", fontsize=20, fontweight="bold")
```

Out[47]: Text(0.5,1,'Skewness calculation')

Out[47]: Text(0.5,1,'Skewness calculation')

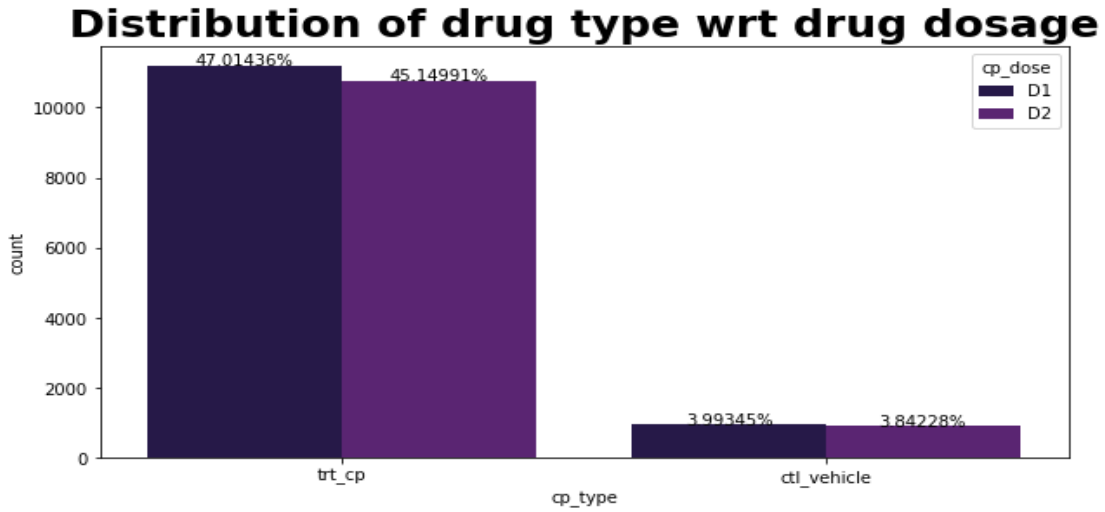


In [50]:
```
# analysis of drug duration and drug drug dosage and plotting results

fig = plt.figure(figsize=(10, 5))
ax = sns.countplot(x="cp_time", hue="cp_dose", data=df_train_features)

for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2,
                height +3,
                '{:1.2f}%'.format(100*height/len(df_train_features)),
                ha="center")

ax.set_title('Distribution of drug duration wrt drug dosage', fontsize=25, fontweight="bold");
```



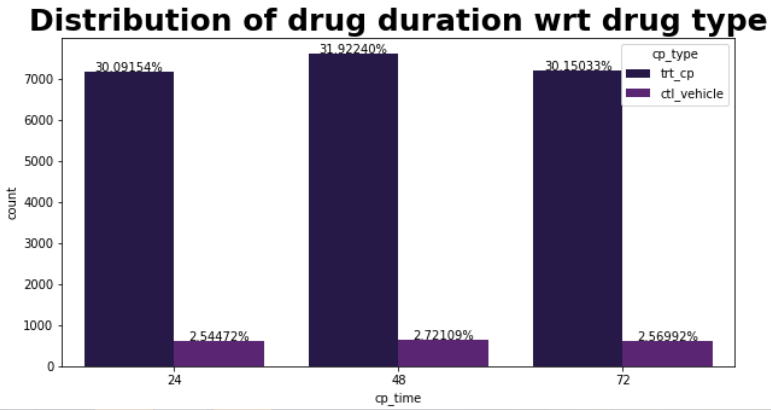
In [54]:

```
# analysis of drug type and drug drug dosage and plotting results

fig = plt.figure(figsize=(10, 5))
ax = sns.countplot(x="cp_type", hue="cp_dose", data=df_train_features)

for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2,
                height +3,
                '{:1.5f}%'.format(100*height/len(df_train_features)),
                ha="center")

ax.set_title('Distribution of drug type wrt drug dosage', fontsize=25, fontweight="bold");
```



In [53]:

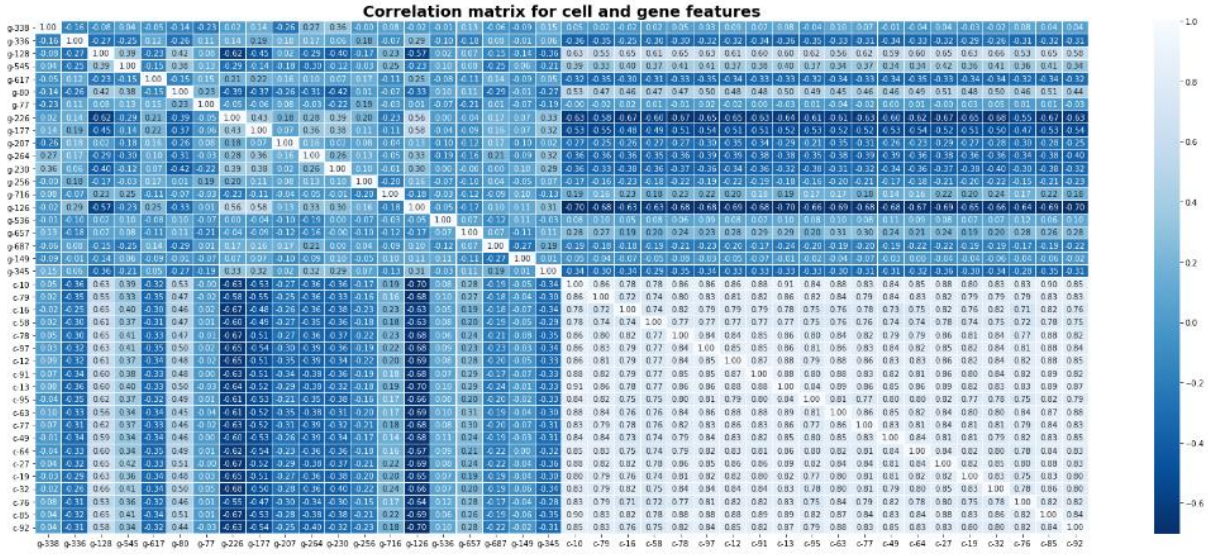
```
# analysis of drug duration and drug type and plotting results

fig = plt.figure(figsize=(10, 5))
ax = sns.countplot(x="cp_time", hue="cp_type", data=df_train_features)

for p in ax.patches:
    height = p.get_height()
    ax.text(p.get_x()+p.get_width()/2,
                height +3,
                '{:1.5f}%'.format(100*height/len(df_train_features)),
                ha="center")

ax.set_title('Distribution of drug duration wrt drug type', fontsize=25, fontweight="bold");
```



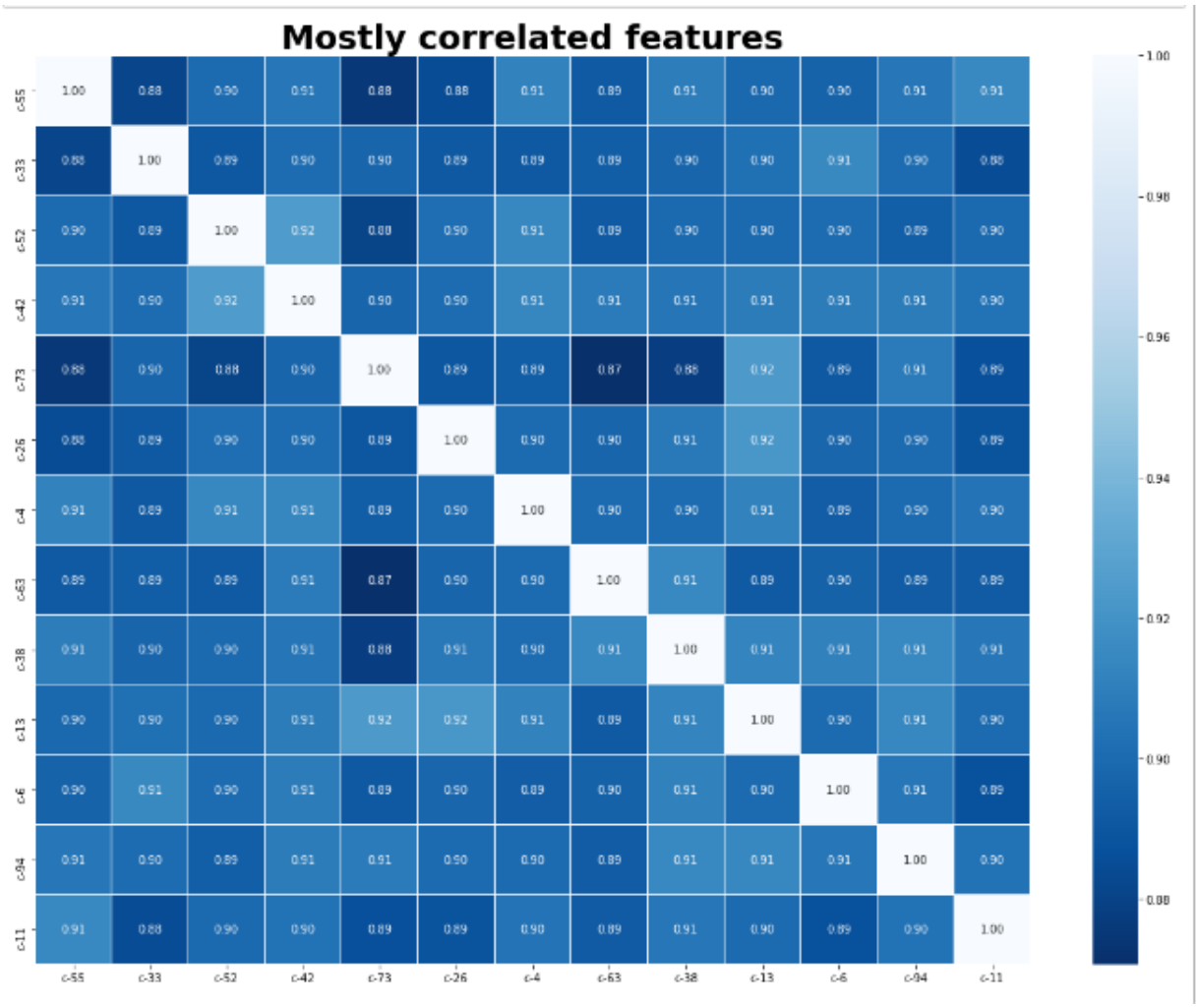
From the above plots, we can infer that the drug type and its effect of duration are having a significant difference between each other.

Now we decided to find out the correlation for all the attributes and then we have filtered out the most significant features by using the correlation matrix

In [58]:
```
# finding correlation matric for cell features and gene features and showing relationship
cell_features = np.random.choice(cell_feature_calculation, 20, replace=False)
gene_features = np.random.choice(gene_features_count_calculation, 20, replace=False)
randomly_selected_features = list(gene_features) + list(cell_features)

plt.figure(figsize=(30, 12))
sns.heatmap(df_train_features[randomly_selected_features].corr(),
            annot=True,
            fmt='0.2f',
            cmap='Blues_r',
            linewidths=0.02,
            cbar=True)

plt.title('Correlation matrix for cell and gene features', fontsize=20, fontweight="bold");
```



In [63]:
```
# finding maximum correlated features and minimum correlated features

corr = df_train_features.iloc[:,1:].corr().abs().unstack().sort_values(kind="quicksort",ascending=False).reset_index()
corr = corr[corr['level_0'] != corr['level_1']]

corr_max = corr.level_0.head(20).tolist()
corr_max = list(set(corr_max))

corr_min = corr.level_0.tail(20).tolist()
corr_min = list(set(corr_min))

plt.figure(figsize=(20, 15))

sns.heatmap(df_train_features[corr_max].corr(),
            annot=True,
            fmt='0.2f',
            cmap='Blues_r',
            linewidths=0.02,
            cbar=True);

plt.title('Mostly correlated features', fontsize=30 , fontweight="bold");
```



CONCLUSION

From the above results, we can infer the following conclusions, and also here are the precise answers for the questions asked in the assignment:

Source of Data: We have picked up this data set from Kaggle which is an open-source community of data scientists and machine learning enthusiasts one of the research going on in the healthcare industry performed by MIT and Harvard for the various tests conducted for the drugs.

The reason behind choosing this dataset: We wanted to do a project in the healthcare industry and this looks like a great challenge as having to improve the performance of the already created algorithm MoA.

Data Exploration: So, we explored the dataset and figured out the relationship between various parameters of drugs, cell features, and gene features as explained in the analysis section.

Conclusion of EDA: We figured out the most significant features with the correlation and relationship among them.

Next Steps: We are planning to do PCA analysis and implement the logistic regression and other modeling techniques later in upcoming assignments**.**

Business Problem? : This data set is taken from a real-time healthcare industry problem statement so we are going to solve and improvise the mechanism of action business problems with the help of this project.

REFERENCES

1. Gaurav Singh, Joelostblom, Rafaelvalle, Apogentus, Phanindra Varma, SM Sivaprakaash (1964, June 01). Plot correlation matrix using pandas. Retrieved November 01, 2020, from https://stackoverflow.com/questions/29432629/plot-correlation-matrix-using-pandas
2. DataVizGuys, Sergey Bushmanov, Niraj, Shayan Shafiq, Predmod, Connor Wilmers (1965, January 01). How to plot a histogram using Matplotlib in Python with a list of data? Retrieved November 01, 2020, from https://stackoverflow.com/questions/33203645/how-to-plot-a-histogram-using-matplotlib-in-python-with-a-list-of-data
3. Koehrsen, W. (2018, March 23). Histograms and Density Plots in Python. Retrieved November 01, 2020, from https://towardsdatascience.com/histograms-and-density-plots-in-python-f6bda88f5ac0
4. Zaric, D. (2020, April 12). Better Heatmaps and Correlation Matrix Plots in Python. Retrieved November 01, 2020, from https://towardsdatascience.com/better-heatmaps-and-correlation-matrix-plots-in-python-41445d0f2bec