Donors Choose Dataset :: Exploratory Data Analysis

Notebook Contents

Understand_the_problem

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

Libraries_Import

In [1]:
```
import logging
logging.basicConfig(filename="Donors_Choose.log",
```

```
                    filemode='w',
                    level=logging.INFO,
                    format="%(asctime)s : %(levelname)s : %(message)s")

try :
    logging.info("#### Packages import ####")
    ## Some basic libaries
    import os
    import sys
    import re          # Tutorial about Python regular expressions: https://pymotw.c
    import string
    import shutil
    import warnings
    import pickle
    import sqlite3
    from tqdm import tqdm
    from collections import Counter

    ## Data Pre-processing Libraries
    import numpy as np
    import pandas as pd
    from prettytable import PrettyTable               # http://zetcode.com/python/pr

    ## Visualization Libraries
    import matplotlib.pyplot as plt
    from matplotlib.colors import ListedColormap

    ### Visualization :: Seaborn
    import seaborn as sns
except ImportError as ie:
    # Output expected ImportErrors
    logging.error(msg=ie.__class__.__name__  + " :: Missing Package --> " + ie.name)
except Exception as exception:
    # Output unexpected Exceptions
    logging.info("#### Exceptions other than ModuleImportError ####")
    logging.log(msg=(exception, False))
    logging.log(msg=exception.__class__.__name__ + " :: " + exception.name)

%matplotlib inline
```

In [2]:
```
## Some display settings
pd.set_option('display.max_rows',105)
pd.set_option('display.max_columns',50)
```

In [3]:
```
## Global Variables
lbl_dict = {'family':'Calibri','size':18,'style':'oblique','color':'k'}
ttl_dict = {'family':'Calibri','size':21,'style':'oblique','color':'magenta'}
wdg_dict = {'linewidth': 1, 'edgecolor': 'black'}
```

Reading_Data

In [4]:
```
dc_train_df = pd.read_csv('Datasets/train_data.csv',index_col=0).reset_index(drop=Tr
dc_res_df = pd.read_csv('Datasets/resources.csv')
```

In [5]:
```
print("Number of data points in train data", (format(dc_train_df.shape[0],',d'),dc_t
print('-'*50)
print("The attributes of train data :", dc_train_df.columns.values)
```

```
Number of data points in train data ('109,248', 16)
--------------------------------------------------
The attributes of train data : ['id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
```

```
'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
'project_essay_4' 'project_resource_summary'
'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [6]:
```
dc_train_df.head(4)
```

Out[6]:

	id	teacher_id	teacher_prefix	school_state	project_submitted_datet
0	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:4
1	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:2
2	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:0
3	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:1

In [7]:
```
print("Number of data points in resource data", (format(dc_res_df.shape[0],',d'),dc_
print('-'*50)
print("The attributes of resource data :", dc_res_df.columns.values)
```

```
Number of data points in resource data ('1,541,272', 4)
--------------------------------------------------
The attributes of resource data : ['id' 'description' 'quantity' 'price']
```

In [8]:
```
dc_res_df.head()
```

Out[8]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95

So, the topic related dataset has around 100k records, whereas, resources dataset has around 1.5 million records. This means that we can have multiple resources requirement for a project.

Univariate_Analysis

Q: How many projects were approved or rejected?

In [9]:
```
dc_train_df['project_is_approved'].value_counts()
```

Out[9]:
```
1    92706
```

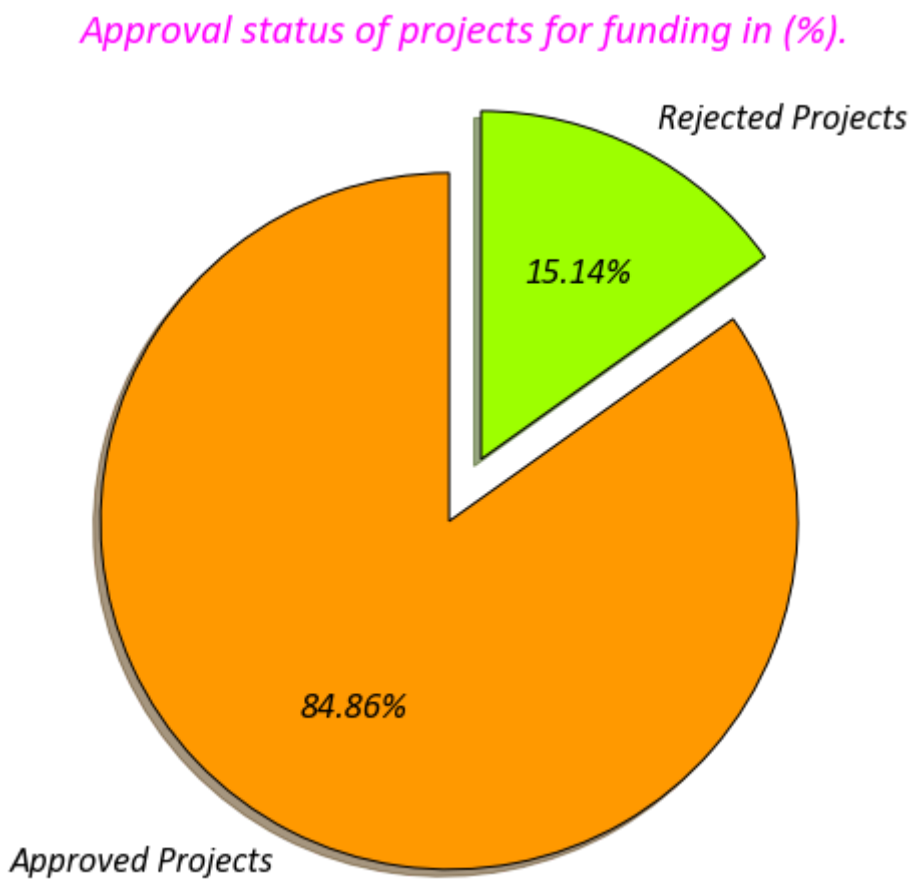
```
0    16542
Name: project_is_approved, dtype: int64
```

In [10]:
```
tot_subms = dc_train_df.shape[0]
tot_apprv_subms = dc_train_df['project_is_approved'].value_counts()[1]
tot_rejec_subms = dc_train_df['project_is_approved'].value_counts()[0]
apprv_percnt = np.divide((1.0*tot_apprv_subms),tot_subms)*100.0
rejec_percnt = np.divide((1.0*tot_rejec_subms),tot_subms)*100.0
```

In [11]:
```
print("Number of projects that are approved for funding --> {:.3f}%".format(apprv_pe
print("Number of projects that are not approved for funding --> {:.3f}%".format(reje
```

```
Number of projects that are approved for funding --> 84.858%
Number of projects that are not approved for funding --> 15.142%
```

In [12]:
```
with plt.style.context('seaborn-bright'):
    plt.figure(figsize=(10,8))
    plt.pie(x=[apprv_percnt,rejec_percnt],
            autopct=lambda pct : "{:.2f}%".format(pct),
            colors=sns.color_palette('gist_rainbow'),
            shadow=True,
            labels=['Approved Projects','Rejected Projects'],
            explode=[0.1,0.1],
            startangle = 90,
            wedgeprops = wdg_dict,
            textprops = lbl_dict)
    plt.title('Approval status of projects for funding in (%).',fontdict=ttl_dict,lo
```



Clearly, there is a huge gap b/w the approved and rejected projects. This states that majority of the submissions were positively moved ahead with funding.

Attribute-1

- **School State**

In [13]:
```
## Thanks to Statistics Canada adding the state names :: https://www23.statcan.gc.ca
us_states_codes = pd.read_csv('Datasets/US_States_with_codes_abbrev.csv',skiprows=1,
```

In [14]:
```
us_states_codes.head()
```

Out[14]:

	State	Alpha code
0	Alabama	AL
1	Alaska	AK
2	Arizona	AZ
3	Arkansas	AR
4	California	CA

In [15]:
```
states_apprv_rejec = pd.DataFrame(dc_train_df.groupby(['project_is_approved','school
rename(columns={'id':'num_of_projects'})
```

In [16]:
```
states_apprv_rejec['state_codes_names'] = states_apprv_rejec['school_state'].\
                                          apply(lambda code: str(us_states_codes[us_st
                                               .replace("[",'').replace("]",'').replac
```

In [17]:
```
states_apprv_rejec['state_codes_names'] = states_apprv_rejec['school_state']+','+sta
```

In [18]:
```
tot_state_subms = dc_train_df['school_state'].value_counts().to_dict()
```

In [19]:
```
states_apprv_rejec['tot_subms'] = states_apprv_rejec['school_state'].apply(lambda va
```

In [20]:
```
states_apprv_rejec['perc_subms'] = states_apprv_rejec[['num_of_projects','tot_subms'
                                   apply(lambda row: np.round((row['num_of_projects
```

In [21]:
```
states_apprv_rejec['state_names'] = states_apprv_rejec['state_codes_names'].apply(la
```

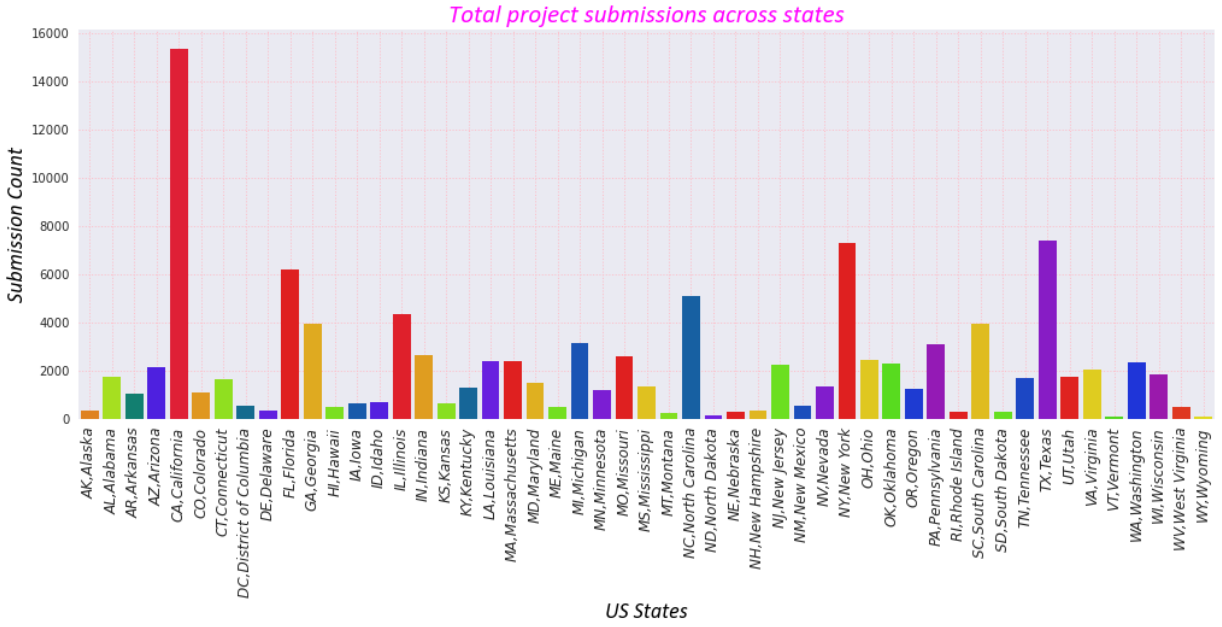
In [22]:
```
states_apprv_rejec.head()
```

Out[22]:

	project_is_approved	school_state	num_of_projects	state_codes_names	tot_subms	perc_subms	st
0	0	AK	55	AK,Alaska	345	15.94	
1	0	AL	256	AL,Alabama	1762	14.53	
2	0	AR	177	AR,Arkansas	1049	16.87	
3	0	AZ	347	AZ,Arizona	2147	16.16	
4	0	CA	2183	CA,California	15388	14.19	

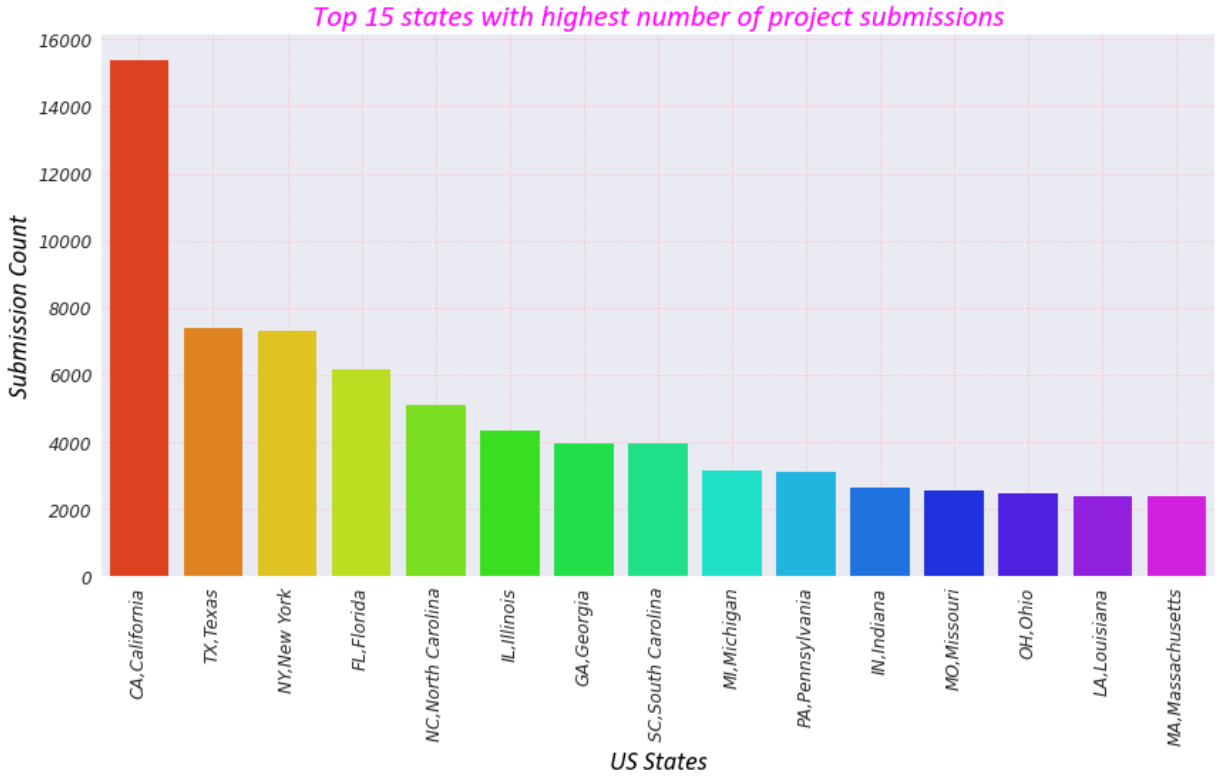
In [23]:
```
with plt.style.context('seaborn'):
    plt.figure(figsize=(14,7))
    sns.barplot(x='state_codes_names',y='tot_subms',data=states_apprv_rejec[states_a
                palette='prism')
    plt.xlabel("US States",fontdict=lbl_dict)
    plt.ylabel("Submission Count",fontdict=lbl_dict)
    plt.grid(linestyle=':',color='pink',which='major')
    plt.xticks(rotation=90,style='oblique',size=12)
```

```
    plt.title("Total project submissions across states",fontdict=ttl_dict)
    plt.tight_layout(pad=0.5,h_pad=0.3)
```



Here, California has a very high numbers of proposals submissions followed by New York, Texas, Florida and others. On the other hand, Vermont, Wyoming and North Dakota are the states which received the least numbers of entires.

In [24]:
```
with plt.style.context('seaborn'):
    plt.figure(figsize=(14,7))
    sns.barplot(x='state_codes_names',y='tot_subms',
                data=states_apprv_rejec[states_apprv_rejec['project_is_approved']==0
                palette='gist_rainbow')
    plt.xlabel("US States",fontdict=lbl_dict)
    plt.ylabel("Submission Count",fontdict=lbl_dict)
    plt.grid(linestyle=':',color='pink',which='major')
    plt.xticks(rotation=90,style='oblique',size=12)
    plt.yticks(style='oblique',size=12)
    plt.title("Top 15 states with highest number of project submissions",fontdict=tt
```

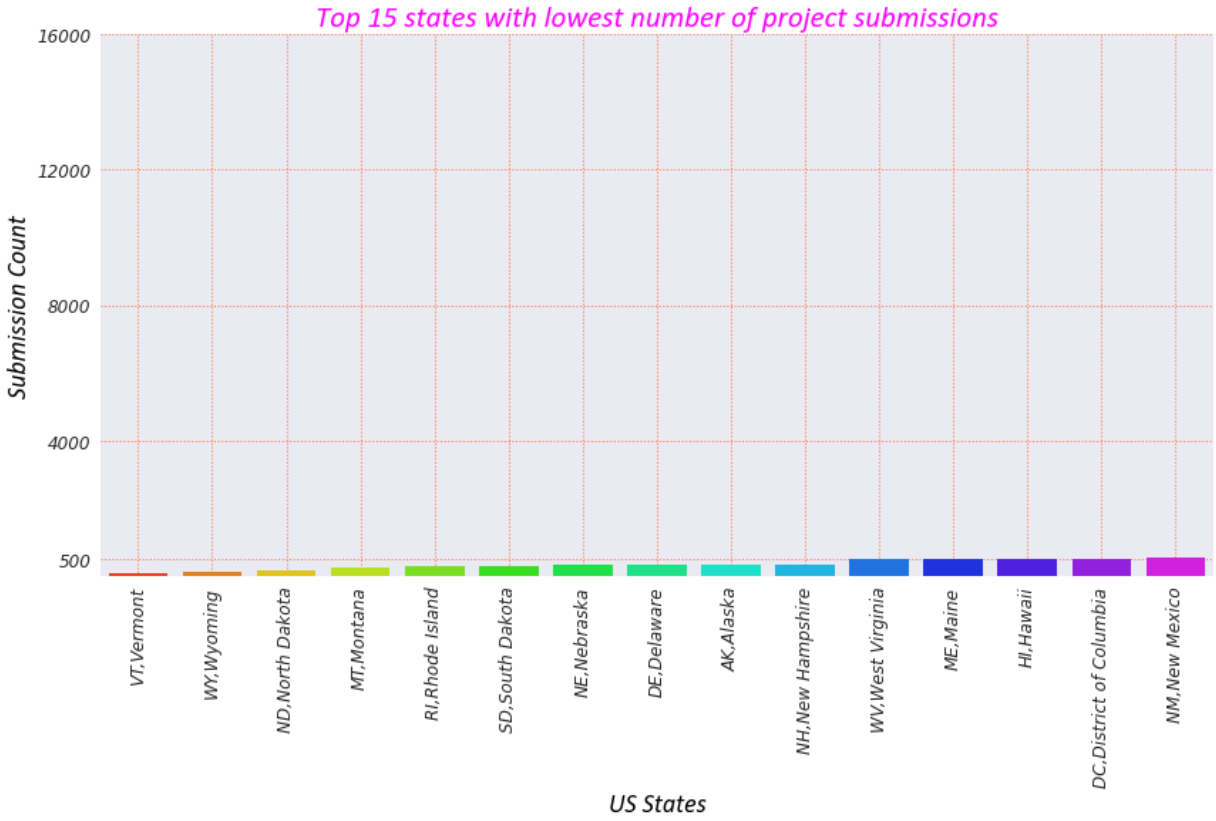


The above plot gives us the top-15 states with the highest number of submissions. California holds almost double submissions as compared to Texas which has the second most highest submissions. There is a very sharp fall in the submissions as we move towards right in the graph.

In [25]:
```
print("*****Top 15 states with highest number of project submissions*****")
data_tab = states_apprv_rejec[states_apprv_rejec['project_is_approved']==0].sort_val
data_tab.reset_index(drop=True,inplace=True)
x = PrettyTable()
x.field_names = ["Seq.","States/Codes", "Submissions"]
for i in range(0,15,1):
    x.add_row([i,data_tab.iloc[i,0],data_tab.iloc[i,1]])
print(x)
```

```
*****Top 15 states with highest number of project submissions*****
+------+-------------------+-------------+
| Seq. |    States/Codes   | Submissions |
+------+-------------------+-------------+
|  0   |   CA,California   |    15388    |
|  1   |      TX,Texas     |     7396    |
|  2   |    NY,New York    |     7318    |
|  3   |     FL,Florida    |     6185    |
|  4   | NC,North Carolina |     5091    |
|  5   |    IL,Illinois    |     4350    |
|  6   |     GA,Georgia    |     3963    |
|  7   | SC,South Carolina |     3936    |
|  8   |    MI,Michigan    |     3161    |
|  9   |  PA,Pennsylvania  |     3109    |
|  10  |     IN,Indiana    |     2620    |
|  11  |    MO,Missouri    |     2576    |
|  12  |      OH,Ohio      |     2467    |
|  13  |    LA,Louisiana   |     2394    |
|  14  |  MA,Massachusetts |     2389    |
+------+-------------------+-------------+
```

In [26]:
```
with plt.style.context('seaborn'):
    plt.figure(figsize=(14,7))
    sns.barplot(x='state_codes_names',y='tot_subms',
                data=states_apprv_rejec[states_apprv_rejec['project_is_approved']==0
                palette='gist_rainbow')
    plt.xlabel("US States",fontdict=lbl_dict)
    plt.ylabel("Submission Count",fontdict=lbl_dict)
    plt.grid(linestyle=':',color='coral',which='major')
    plt.xticks(rotation=90,style='oblique',size=12)
    plt.title("Top 15 states with lowest number of project submissions",fontdict=ttl
    plt.yticks([500,4000,8000,12000,16000])
    plt.yticks(style='oblique',size=12)
```



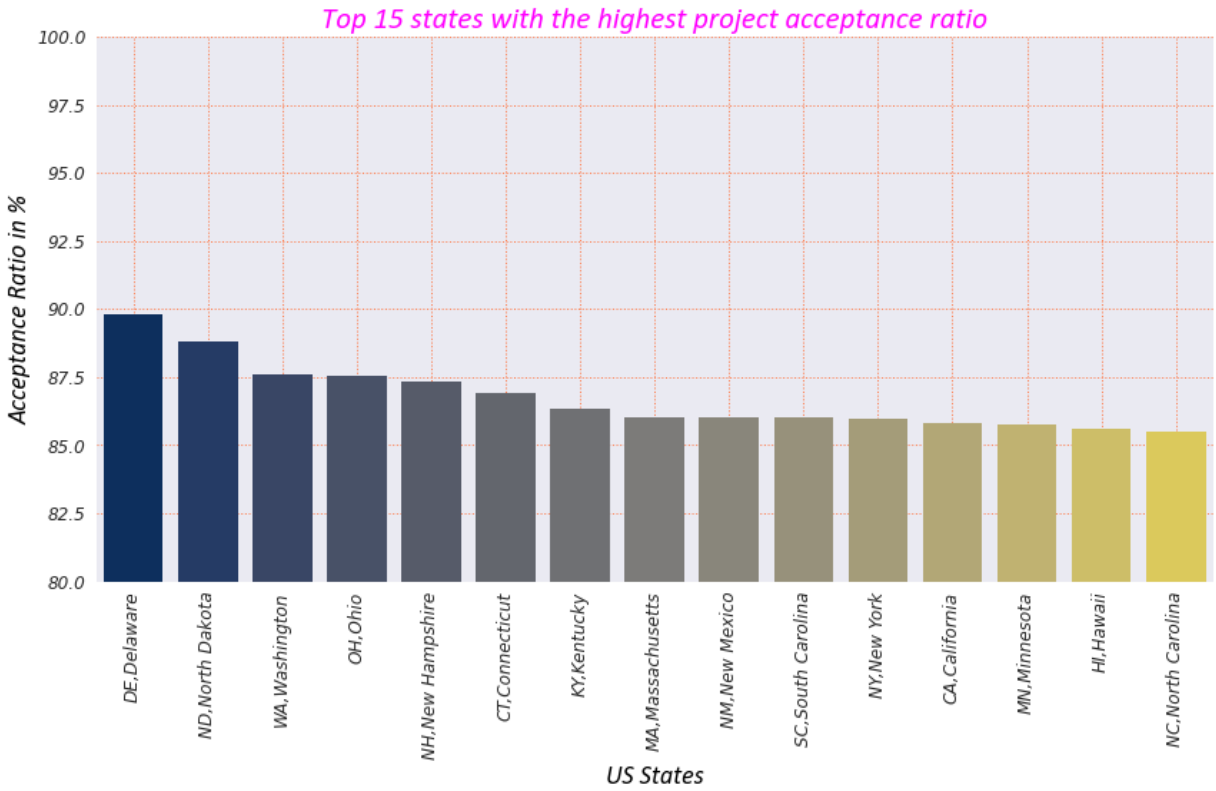
The above plot gives us the top-15 states with the lowest number of submissions. Majority of these holds less than 500 submissions, it would be good to know if any of these states has a high acceptance percentage.

In [27]:
```
print("*****Top 15 states with lowest number of project submissions*****")
data_tab = states_apprv_rejec[states_apprv_rejec['project_is_approved']==0].sort_val
data_tab.reset_index(drop=True,inplace=True)
x = PrettyTable()
x.field_names = ["Seq.","States/Codes", "Submissions"]
for i in range(0,15,1):
    x.add_row([i,data_tab.iloc[i,0],data_tab.iloc[i,1]])
print(x)
```

```
*****Top 15 states with lowest number of project submissions*****
+------+-------------------------+-------------+
| Seq. |       States/Codes      | Submissions |
+------+-------------------------+-------------+
|  0   |        VT,Vermont       |      80     |
|  1   |        WY,Wyoming       |      98     |
|  2   |     ND,North Dakota     |     143     |
|  3   |        MT,Montana       |     245     |
|  4   |     RI,Rhode Island     |     285     |
|  5   |     SD,South Dakota     |     300     |
|  6   |       NE,Nebraska       |     309     |
|  7   |       DE,Delaware       |     343     |
|  8   |        AK,Alaska        |     345     |
|  9   |     NH,New Hampshire    |     348     |
|  10  |     WV,West Virginia    |     503     |
|  11  |         ME,Maine        |     505     |
|  12  |        HI,Hawaii        |     507     |
|  13  | DC,District of Columbia |     516     |
|  14  |      NM,New Mexico      |     557     |
+------+-------------------------+-------------+
```

In [28]:
```
with plt.style.context('seaborn'):
    plt.figure(figsize=(14,7))
    sns.barplot(x='state_codes_names',y='perc_subms',
                data=states_apprv_rejec[states_apprv_rejec['project_is_approved']==1
                palette='cividis')
```

```
    plt.xlabel("US States",fontdict=lbl_dict)
    plt.ylabel("Acceptance Ratio in %",fontdict=lbl_dict)
    plt.xticks(rotation=90,style='oblique',size=12)
    plt.yticks(style='oblique',size=12)
    plt.grid(linestyle=':',color='coral',which='major')
    plt.ylim(80,100)
    plt.title("Top 15 states with the highest project acceptance ratio",fontdict=ttl
```



The above plot gives us the top-15 states which holds the highest projects acceptance percentage. So, clearly here the top two states have the lowest submissions but with a high acceptance ratio. California who holds the highest numbers of submissions has approximately 86% of acceptance.

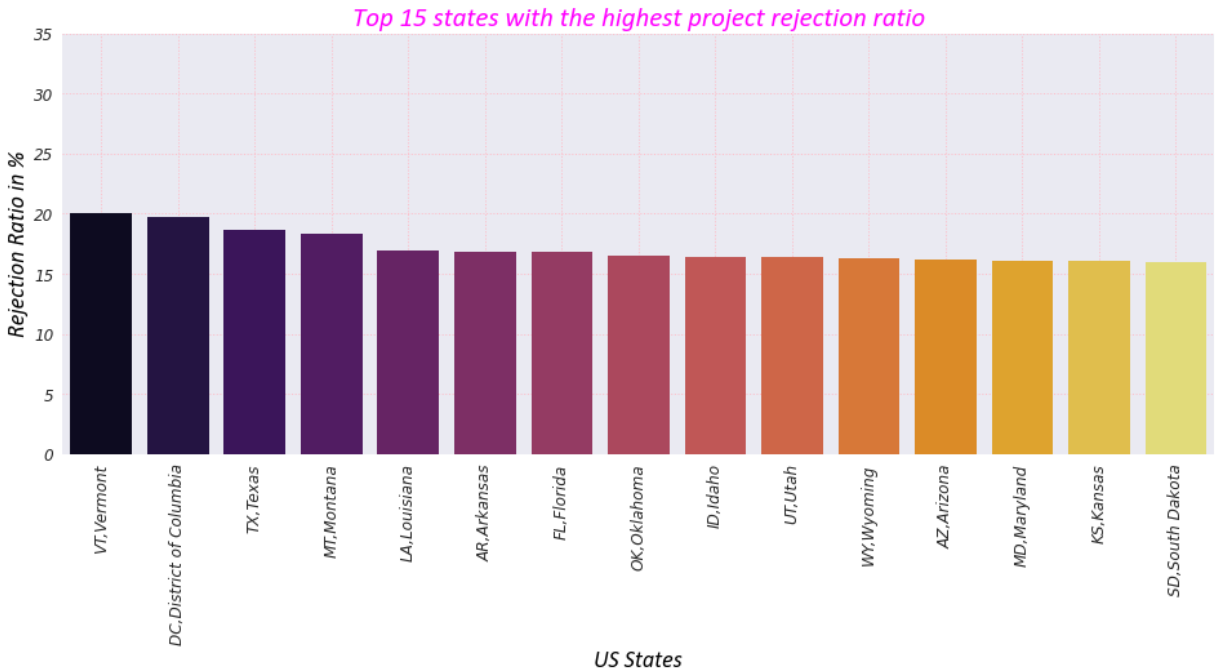
And, Texas which is the second state in terms of submissions is not present in this list.

In [29]:
```
print("*****Top 15 states with the highest project acceptance ratio*****")
data_tab = states_apprv_rejec[states_apprv_rejec['project_is_approved']==1].sort_val
data_tab.reset_index(drop=True,inplace=True)
x = PrettyTable()
x.field_names = ["Seq.","States/Codes","Accepted_Submissions","Total_Submissions","A
for i in range(0,15,1):
    x.add_row([i,data_tab.iloc[i,0],data_tab.iloc[i,1],data_tab.iloc[i,2],data_tab.i
print(x)
```

```
*****Top 15 states with the highest project acceptance ratio*****
+------+------------------+----------------------+------------------+-------------
+
| Seq. |    States/Codes  | Accepted_Submissions | Total_Submissions | Acceptance%
|
+------+------------------+----------------------+------------------+-------------
+
|  0   |    DE,Delaware   |         308          |        343        |     89.8
|
|  1   |  ND,North Dakota |         127          |        143        |    88.81
|
|  2   |   WA,Washington  |         2045         |        2334       |    87.62
|
|  3   |      OH,Ohio     |         2159         |        2467       |    87.52
|
```

4	NH,New Hampshire	304	348	87.36
5	CT,Connecticut	1445	1663	86.89
6	KY,Kentucky	1126	1304	86.35
7	MA,Massachusetts	2055	2389	86.02
8	NM,New Mexico	479	557	86.0
9	SC,South Carolina	3385	3936	86.0
10	NY,New York	6291	7318	85.97
11	CA,California	13205	15388	85.81
12	MN,Minnesota	1036	1208	85.76
13	HI,Hawaii	434	507	85.6
14	NC,North Carolina	4353	5091	85.5

In [30]:
```
with plt.style.context('seaborn'):
    plt.figure(figsize=(16,6))
    sns.barplot(x='state_codes_names',y='perc_subms',
                data=states_apprv_rejec[states_apprv_rejec['project_is_approved']==0
                palette='inferno')
    plt.xlabel("US States",fontdict=lbl_dict)
    plt.ylabel("Rejection Ratio in %",fontdict=lbl_dict)
    plt.xticks(rotation=90,style='oblique',size=12)
    plt.yticks(style='oblique',size=12)
    plt.grid(linestyle=':',color='pink',which='major')
    plt.ylim(0,35)
    plt.title("Top 15 states with the highest project rejection ratio",fontdict=ttl_
```



The above plot gives us the top-15 states which holds the highest projects rejection ratio. Here, Vermont which holds the least number of submissions has the highest rejection ratio of 20%. Texas the state with 2nd most highest submissions holds a rejection ratio of approx. 19%.

And, Florida which also holds some high number of submissions has a rejection % of approx. 17%.

In [31]:
```
print("*****Top 15 states with the highest project rejection ratio*****")
data_tab = states_apprv_rejec[states_apprv_rejec['project_is_approved']==0].sort_val
data_tab.reset_index(drop=True,inplace=True)
x = PrettyTable()
x.field_names = ["Seq.","States/Codes","Rejected_Submissions","Total Submissions","R
for i in range(0,15,1):
    x.add_row([i,data_tab.iloc[i,0],data_tab.iloc[i,1],data_tab.iloc[i,2],data_tab.i
print(x)
```

```
*****Top 15 states with the highest project rejection ratio*****
+------+-------------------------+---------------------+------------------+-------
-----+
| Seq. |       States/Codes      | Rejected_Submissions | Total Submissions | Reject
ion% |
+------+-------------------------+---------------------+------------------+-------
-----+
|  0   |        VT,Vermont       |          16          |         80        |    20.
0    |
|  1   | DC,District of Columbia |         102          |        516        |   19.7
7    |
|  2   |         TX,Texas        |         1382         |        7396       |   18.6
9    |
|  3   |        MT,Montana       |          45          |        245        |   18.3
7    |
|  4   |       LA,Louisiana      |         404          |        2394       |   16.8
8    |
|  5   |       AR,Arkansas       |         177          |        1049       |   16.8
7    |
|  6   |        FL,Florida       |         1041         |        6185       |   16.8
3    |
|  7   |       OK,Oklahoma       |         376          |        2276       |   16.5
2    |
|  8   |         ID,Idaho        |         114          |        693        |   16.4
5    |
|  9   |         UT,Utah         |         283          |        1731       |   16.3
5    |
|  10  |        WY,Wyoming       |          16          |         98        |   16.3
3    |
|  11  |        AZ,Arizona       |         347          |        2147       |   16.1
6    |
|  12  |       MD,Maryland       |         244          |        1514       |   16.1
2    |
|  13  |        KS,Kansas        |         102          |        634        |   16.0
9    |
|  14  |     SD,South Dakota     |          48          |        300        |    16.
0    |
+------+-------------------------+---------------------+------------------+-------
-----+
```

Data Analysis Functions

In [32]:
```
def analysis_data_prep(df_obj,tgt_val,indp_val,trunc_flg):
    """
    Description: This function is created for performing the univariate data analysi

    Input parameters: It accepts below inputs:
        1. `df_obj`: Pandas DataFrame
                Dataframe containing the entire dataset.
        2. `tgt_val`: str
                Target class attribute name exists in df_obj will be used for groupi
        3. `indp_val`: str
                Feature/Attribute name that exists in df_obj and needs to be analyse
        4. `trunc_flg`: boolean
                This is a flag parameter used for limiting the data if many classes
```

```
    Return: It returns either of the pandas dataframe objects `tmp5` or `tmp`.
        `tmp5`: Returned if trunc_flg is True then 60 records of every target class
                (30 from both top & bottom total submissions)
         else
        `tmp` is returned which contains data from all the feature .
    """
    tmp_df = df_obj.copy(deep=True)
    tmp_df.reset_index(inplace=True)

    # Calculating the total submissions in every feature class
    total_subms = tmp_df[indp_val].value_counts().to_dict()

    # Generating the multi-index to get the zero count as well if record doesn't exi
    mul_idx = pd.MultiIndex.from_product([tmp_df[tgt_val].unique(),tmp_df[indp_val].
    tmp = pd.DataFrame(tmp_df.groupby([tgt_val,indp_val])['index'].count()).reindex(
    .reset_index().rename(columns={'index':'count'}).copy(deep=True)

    # Generating the total submissions column
    tmp['tot_subms'] = tmp[indp_val].apply(lambda val: total_subms.get(val))

    # Generating the percentage submissions column
    tmp['perc_subms'] = tmp[['count','tot_subms']].apply(lambda row: np.round((row['

    # trunc_flg is used if feature variable contains numerous categories
    if trunc_flg:
        tmp1 = tmp[tmp[tgt_val]==0].sort_values(by='tot_subms',ascending=False).iloc
        tmp2 = tmp[tmp[tgt_val]==0].sort_values(by='tot_subms',ascending=True).iloc[
        tmp3 = tmp[tmp[tgt_val]==1].sort_values(by='tot_subms',ascending=False).iloc
        tmp4 = tmp[tmp[tgt_val]==1].sort_values(by='tot_subms',ascending=True).iloc[
        tmp5 = pd.concat([tmp1,tmp2,tmp3,tmp4],axis=0).reset_index(drop=True)
        del tmp_df
        return tmp5
    else:
        del tmp_df
        return tmp

def univariate_analysis(df_obj,tgt_val,indp_val,xlabel,ylabel,t_flg=False):
    """
    Description: This function is created for plotting the graphs.

    Input parameters: It accepts below inputs:
        1. `df_obj`: Pandas DataFrame
                Dataframe containing the entire dataset.
        2. `tgt_val`: str
                Target class attribute name exists in df_obj will be used for groupi
        3. `indp_val`: str
                Feature/Attribute name that exists in df_obj and needs to be analyse
        4. `xlabel`: str
                X-axis label used in plotting.
        5. `ylabel`: str
                Y-axis label used in plotting.
        6. `t_flg`: boolean
                This is a flag parameter used for limiting the data if many classes

    Return: None
    """
    # Calling the analysis data prep function
    df = analysis_data_prep(df_obj,tgt_val,indp_val,t_flg)

    # Limiting the features classes for plotting
    unique_categories = int(df[df[tgt_val]==0][indp_val].nunique())
    if unique_categories > 20:
        thresh_rec = 16
    else:
```

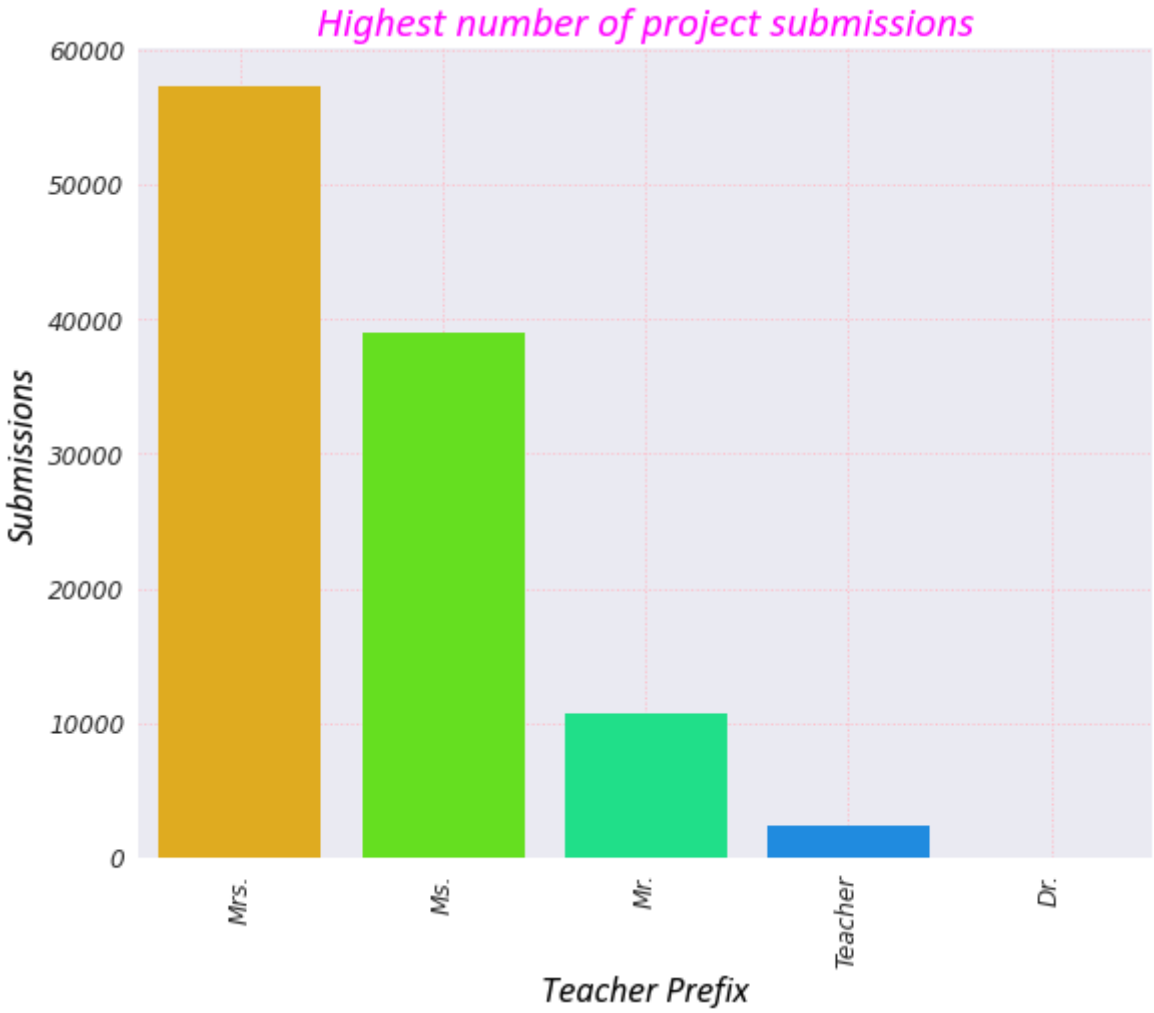
```
    thresh_rec = unique_categories

def fig_size(th_flag, diff_plot=False):
    """
    Description: This function is created for providing the figure size based on

    Inputs: It accepts two parameters:
        1. `th_flag`: boolean
            This is a flag parameter used for limiting the data if many classes
        2. `diff_plot`: boolean
            This is another flag parameter for providing a different size if dat

    Return:
        1. `f_size`: tuple
            This is the figure size to be used for plotting.
    """
    if th_flag == True and diff_plot==True:
        f_size = (14,16)
        return f_size
    elif th_flag == True and diff_plot==False:
        f_size = (14,7)
        return f_size
    else:
        f_size = (8,7)
        return f_size

## Plot-1 :: Displaying "Highest number of project submissions"
with plt.style.context('seaborn'):
    plt.figure(figsize=fig_size(t_flg))
    sns.barplot(x=indp_val,y='tot_subms',data=df[df[tgt_val]==0].sort_values(by=
    plt.xlabel(xlabel,fontdict=lbl_dict)
    plt.ylabel(ylabel,fontdict=lbl_dict)
    plt.grid(linestyle=':',color='pink',which='major')
    plt.xticks(rotation=90,style='oblique',size=12)
    plt.yticks(style='oblique',size=12)
    plt.title("Highest number of project submissions",fontdict=ttl_dict)
    plt.minorticks_on()
    plt.tight_layout(pad=0.5,h_pad=0.3)
plt.show()

#### 1.1 :: Printing outcome values in a prettytable
print('\n',"*** Highest number of project submissions ***")

data_tab = df[df[tgt_val]==0].sort_values(by='tot_subms',ascending=False)\
.reset_index(drop=True).iloc[:thresh_rec,:][[indp_val,'tot_subms']].rename(colum
data_tab.reset_index(drop=True,inplace=True)

x = PrettyTable()
x.field_names = ["Seq.",indp_val, "Total_Submissions"]
for i in range(0,thresh_rec,1):
    x.add_row([i,data_tab.iloc[i,0],data_tab.iloc[i,1]])
print(x,'\n')

## Plot-2 :: Displaying the data in a stacked bar plot
x_index = np.arange(df[df[tgt_val]==0].shape[0])
with plt.style.context('seaborn'):
    plt.figure(figsize=fig_size(t_flg,diff_plot=True))
    total = plt.barh(x_index, df[df[tgt_val]==0]['tot_subms'].values)
    approved = plt.barh(x_index, df[df[tgt_val]==1]['count'].values)
    plt.xlabel(ylabel,fontdict=lbl_dict)
    plt.ylabel(xlabel,fontdict=lbl_dict)
    plt.title('% of projects aproved for funding',fontdict=ttl_dict)
    plt.yticks(x_index,list(df[df[tgt_val]==0][indp_val].values),rotation=0,styl
    plt.xticks(style='oblique',size=12)
    plt.legend((total[0], approved[0]), ('Total', 'Approved'))
```

```
    plt.minorticks_on()
    plt.tight_layout(pad=0.5,h_pad=0.3)
plt.show()
print('\n')

## Plot-3 :: Displaying "Highest project acceptance ratio"
with plt.style.context('seaborn'):
    plt.figure(figsize=fig_size(t_flg))
    sns.barplot(x=indp_val,y='perc_subms',data=df[df[tgt_val]==1].sort_values(by
    plt.xlabel(xlabel,fontdict=lbl_dict)
    plt.ylabel("Approved %",fontdict=lbl_dict)
    plt.xticks(rotation=90,style='oblique',size=12)
    plt.yticks(style='oblique',size=12)
    plt.grid(linestyle=':',color='pink',which='major')
    plt.ylim(60,100)
    plt.title("Highest project acceptance ratio",fontdict=ttl_dict)
    plt.minorticks_on()
    plt.tight_layout(pad=0.5,h_pad=0.3)
plt.show()

#### 3.1 :: Printing outcome values in a prettytable
print('\n',"*** Highest project acceptance ratio ***")

data_tab = df[df[tgt_val]==1].sort_values(by='perc_subms',ascending=False)\
.reset_index(drop=True).iloc[:thresh_rec,:][[indp_val,'count','tot_subms','perc_
.rename(columns={'count':'Apprv_Submissions','tot_subms':'Total_Submissions','pe

data_tab.reset_index(drop=True,inplace=True)
x = PrettyTable()
x.field_names = ["Seq.",indp_val,"Apprv_Submissions","Total_Submissions","Apprv_
for i in range(0,thresh_rec,1):
    x.add_row([i,data_tab.iloc[i,0],data_tab.iloc[i,1],data_tab.iloc[i,2],data_t
print(x,'\n')

## Plot-4 :: Displaying "Highest project rejection ratio"
"""Although the plot-3 is an opposite of plot-4 and kind of portray the same inf
   but its always better to plot it explicitly"""
with plt.style.context('seaborn'):
    plt.figure(figsize=fig_size(t_flg))
    sns.barplot(x=indp_val,y='perc_subms',data=df[df[tgt_val]==0].sort_values(by
    plt.xlabel(xlabel,fontdict=lbl_dict)
    plt.ylabel("Rejection %",fontdict=lbl_dict)
    plt.xticks(rotation=90,style='oblique',size=12)
    plt.yticks(style='oblique',size=12)
    plt.grid(linestyle=':',color='pink',which='major')
    plt.ylim(0,50)
    plt.title("Highest project rejection ratio",fontdict=ttl_dict)
    plt.minorticks_on()
    plt.tight_layout(pad=0.5,h_pad=0.3)
plt.show()

#### 4.1 ::  Printing outcome values in a prettytable
print('\n',"*** Highest project rejection ratio ***")

data_tab = df[df[tgt_val]==0].sort_values(by='perc_subms',ascending=False)\
.reset_index(drop=True).iloc[:thresh_rec,:][[indp_val,'count','tot_subms','perc_
.rename(columns={'count':'Reject_Submissions','tot_subms':'Total_Submissions','p

data_tab.reset_index(drop=True,inplace=True)
x = PrettyTable()
x.field_names = ["Seq.",indp_val,"Reject_Submissions","Total_Submissions","Rejec
for i in range(0,thresh_rec,1):
    x.add_row([i,data_tab.iloc[i,0],data_tab.iloc[i,1],data_tab.iloc[i,2],data_t
print(x,'\n')
```

Attribute-2

- **Teacher Prefix**

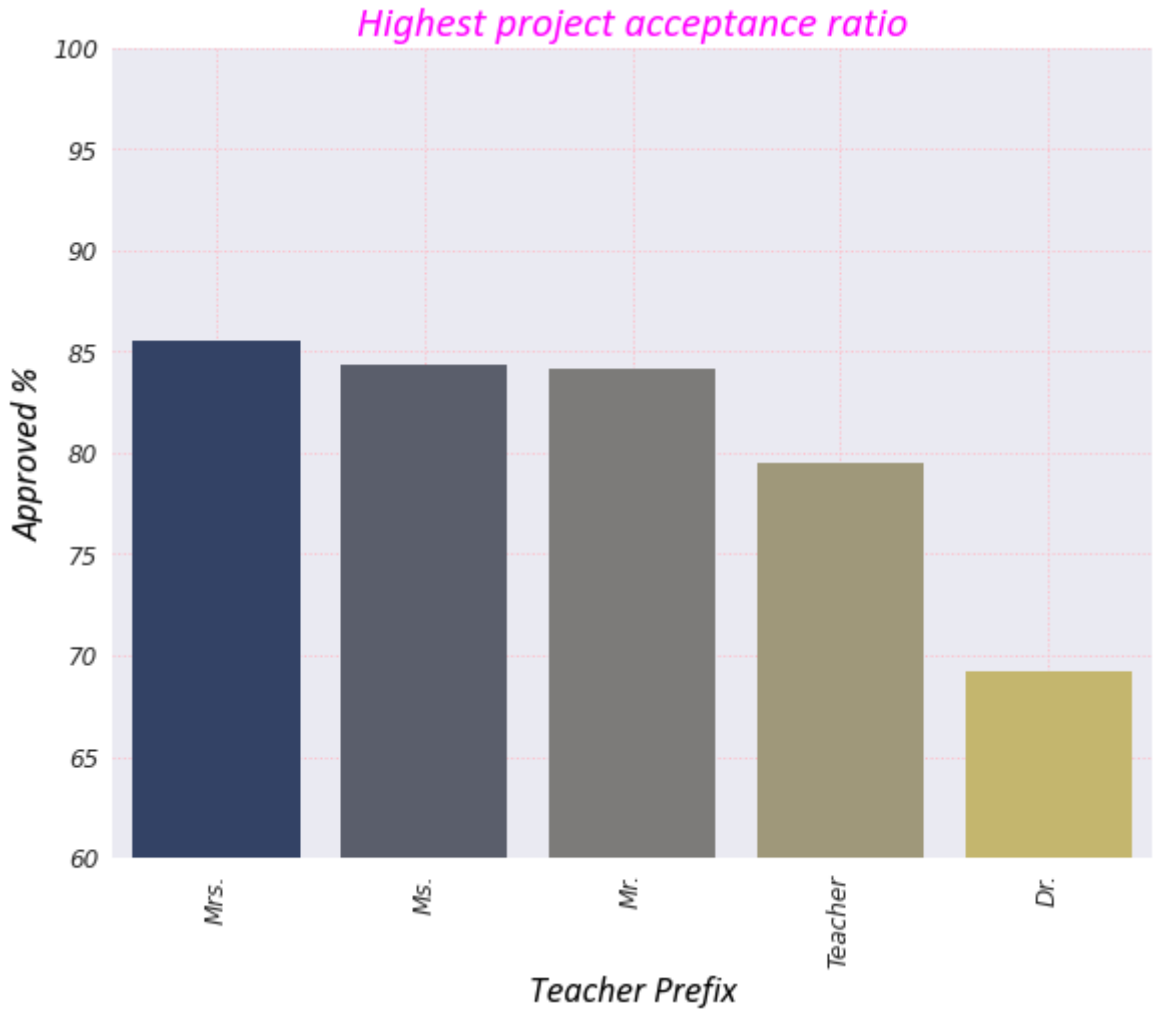
In [33]:
```
univariate_analysis(dc_train_df,'project_is_approved','teacher_prefix','Teacher Pref
```



*** Highest number of project submissions ***

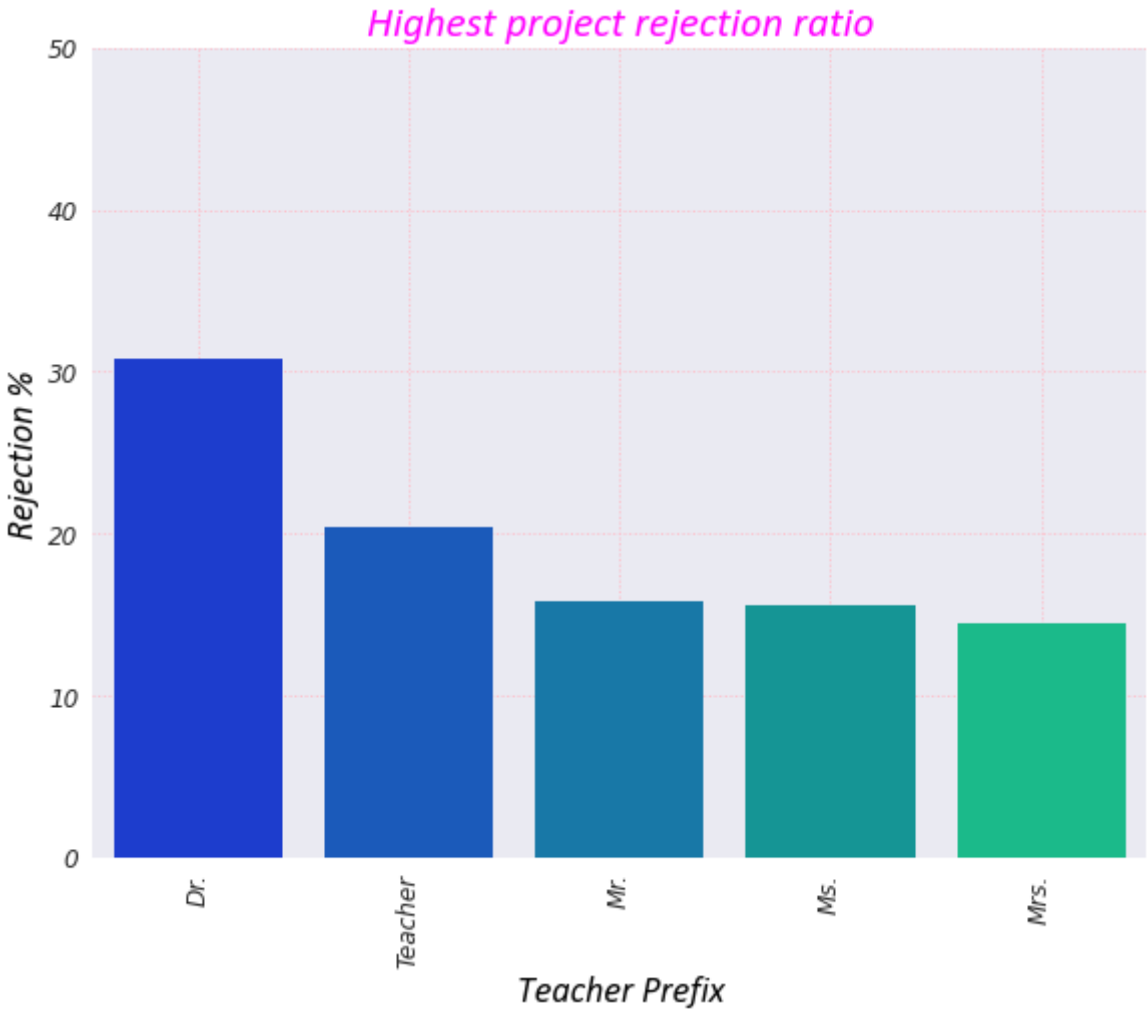
Seq.	teacher_prefix	Total_Submissions
0	Mrs.	57269.0
1	Ms.	38955.0
2	Mr.	10648.0
3	Teacher	2360.0
4	Dr.	13.0


% of projects aproved for funding
Total
Approved
Dr.
nan
Teacher
Ms.
Mr.
Mrs.
Teacher Prefix
0
10000
20000
30000
40000
50000
60000
Submissions


Highest project acceptance ratio
100
95
90
85
80
75
70
65
60
Approved %
Mrs.
Ms.
Mr.
Teacher
Dr.
Teacher Prefix

*** Highest project acceptance ratio ***

Seq.	teacher_prefix	Apprv_Submissions	Total_Submissions	Apprv_%
0	Mrs.	48997	57269.0	85.56
1	Ms.	32860	38955.0	84.35
2	Mr.	8960	10648.0	84.15
3	Teacher	1877	2360.0	79.53
4	Dr.	9	13.0	69.23



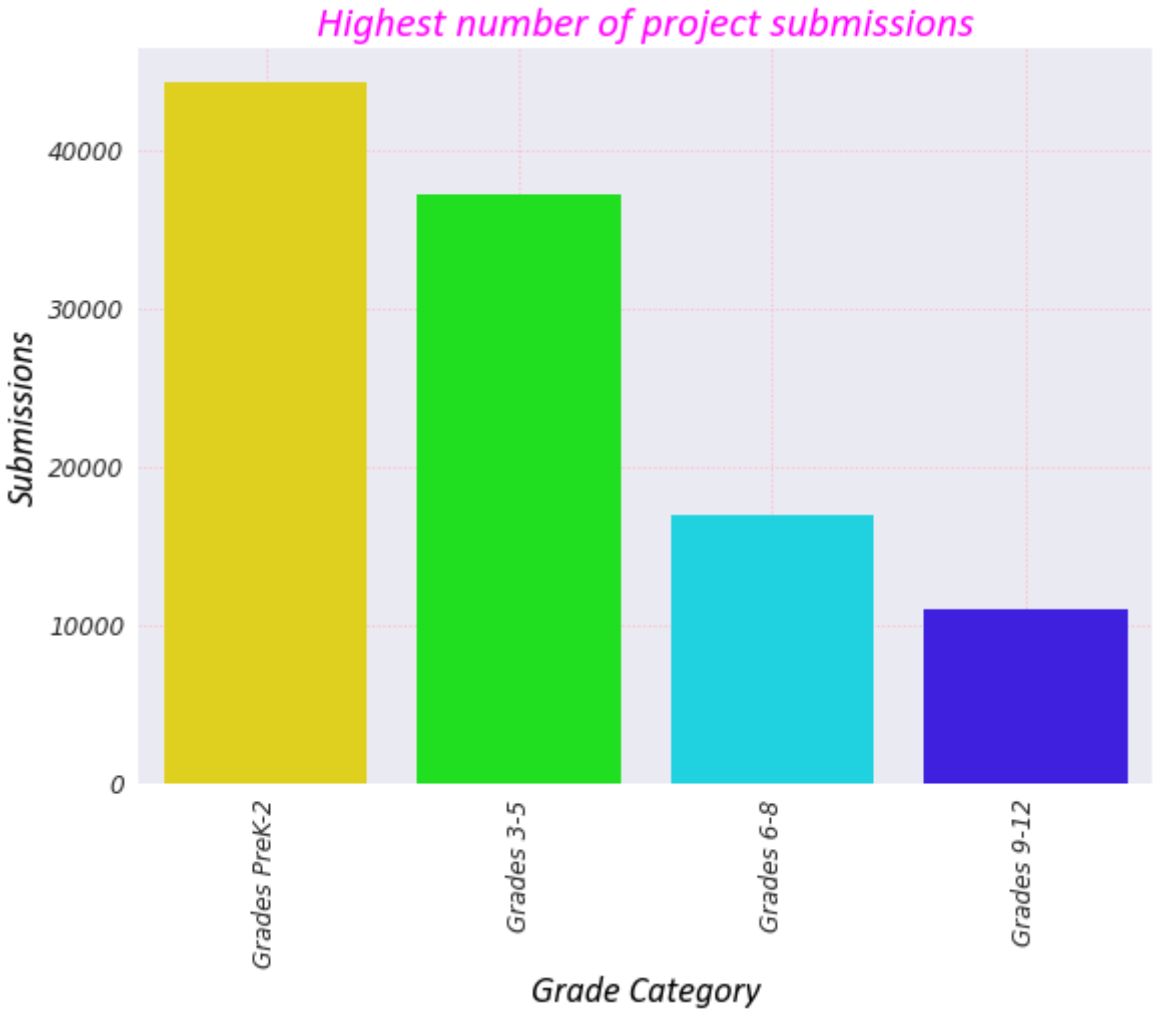
*** Highest project rejection ratio ***

Seq.	teacher_prefix	Reject_Submissions	Total_Submissions	Reject_%
0	Dr.	4	13.0	30.77
1	Teacher	483	2360.0	20.47
2	Mr.	1688	10648.0	15.85
3	Ms.	6095	38955.0	15.65
4	Mrs.	8272	57269.0	14.44

Attribute-3

- **Project Grade Category**

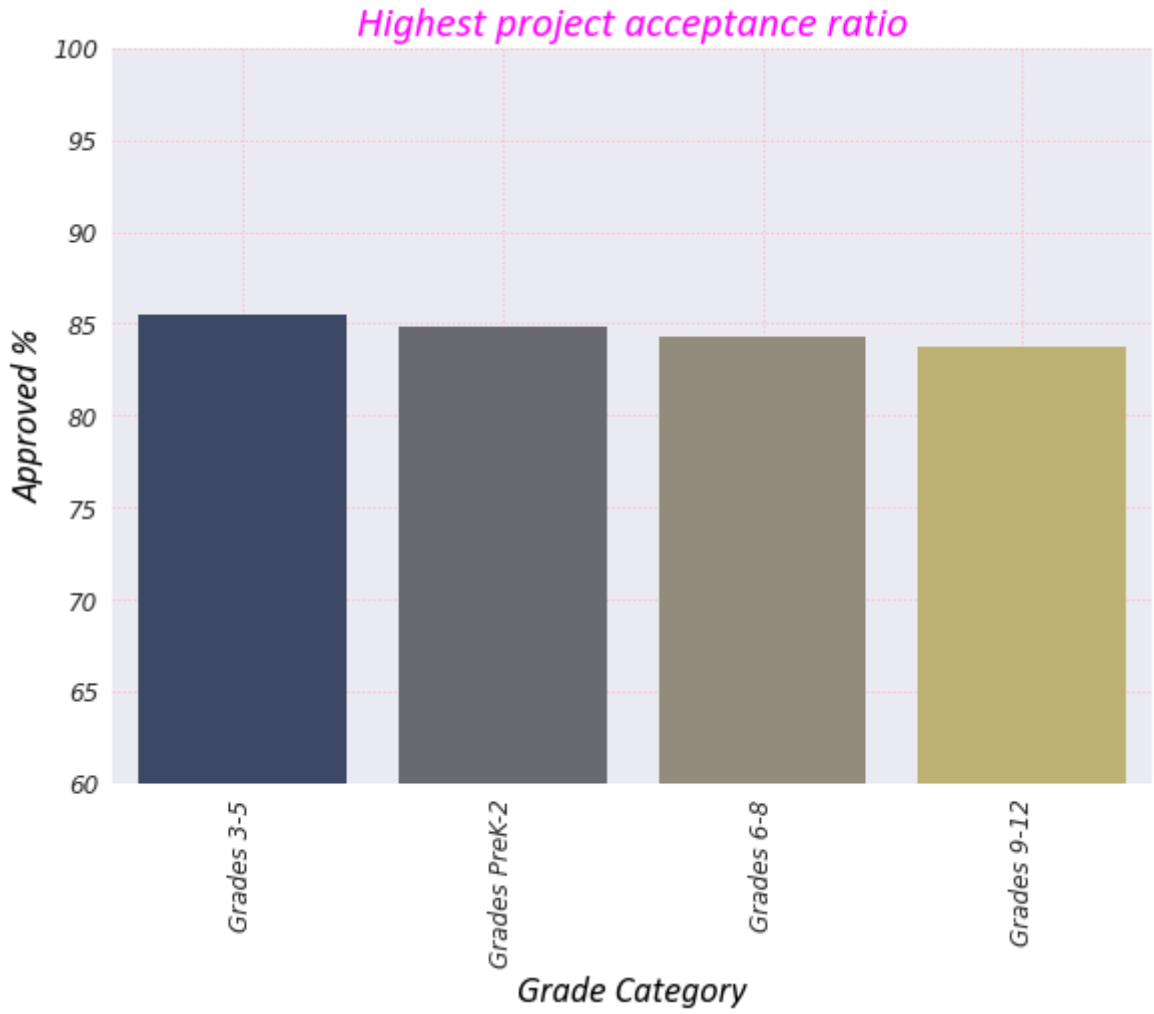
In [34]:
```
univariate_analysis(dc_train_df,'project_is_approved','project_grade_category','Grad
```



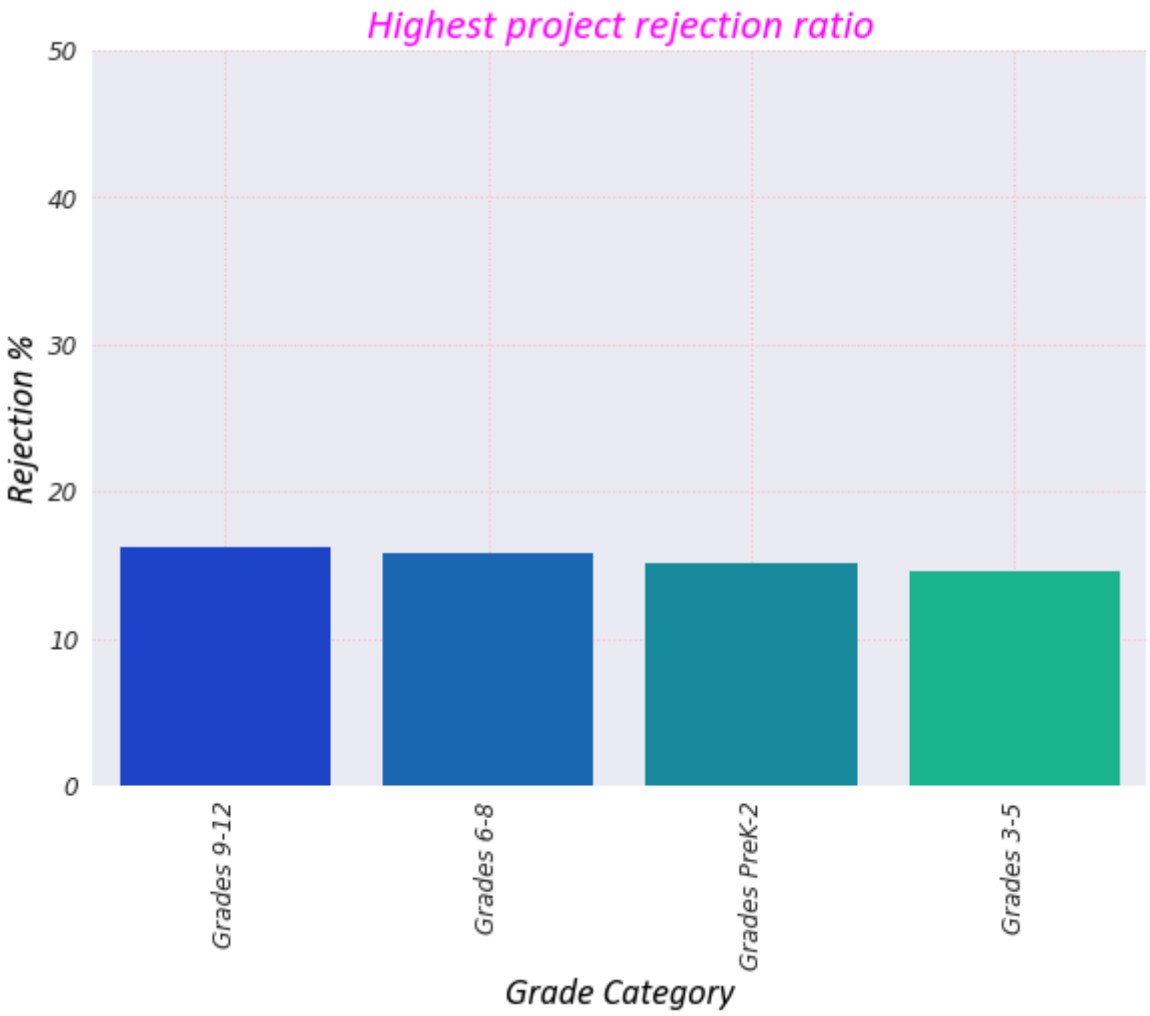
*** Highest number of project submissions ***

Seq.	project_grade_category	Total_Submissions
0	Grades PreK-2	44225
1	Grades 3-5	37137
2	Grades 6-8	16923
3	Grades 9-12	10963


% of projects aproved for funding
Total
Approved
Grades 9-12
Grades 3-5
Grades 6-8
Grades PreK-2
Grade Category
0
10000
20000
30000
40000
Submissions


Highest project acceptance ratio
100
95
90
85
80
75
70
65
60
Approved %
Grades 3-5
Grades PreK-2
Grades 6-8
Grades 9-12
Grade Category

```
 *** Highest project acceptance ratio ***
+------+------------------------+-------------------+-------------------+---------+
| Seq. | project_grade_category | Apprv_Submissions | Total_Submissions | Apprv_% |
+------+------------------------+-------------------+-------------------+---------+
|  0   |       Grades 3-5       |       31729       |       37137       |  85.44  |
|  1   |     Grades PreK-2      |       37536       |       44225       |  84.88  |
|  2   |       Grades 6-8       |       14258       |       16923       |  84.25  |
|  3   |      Grades 9-12       |        9183       |       10963       |  83.76  |
+------+------------------------+-------------------+-------------------+---------+
```

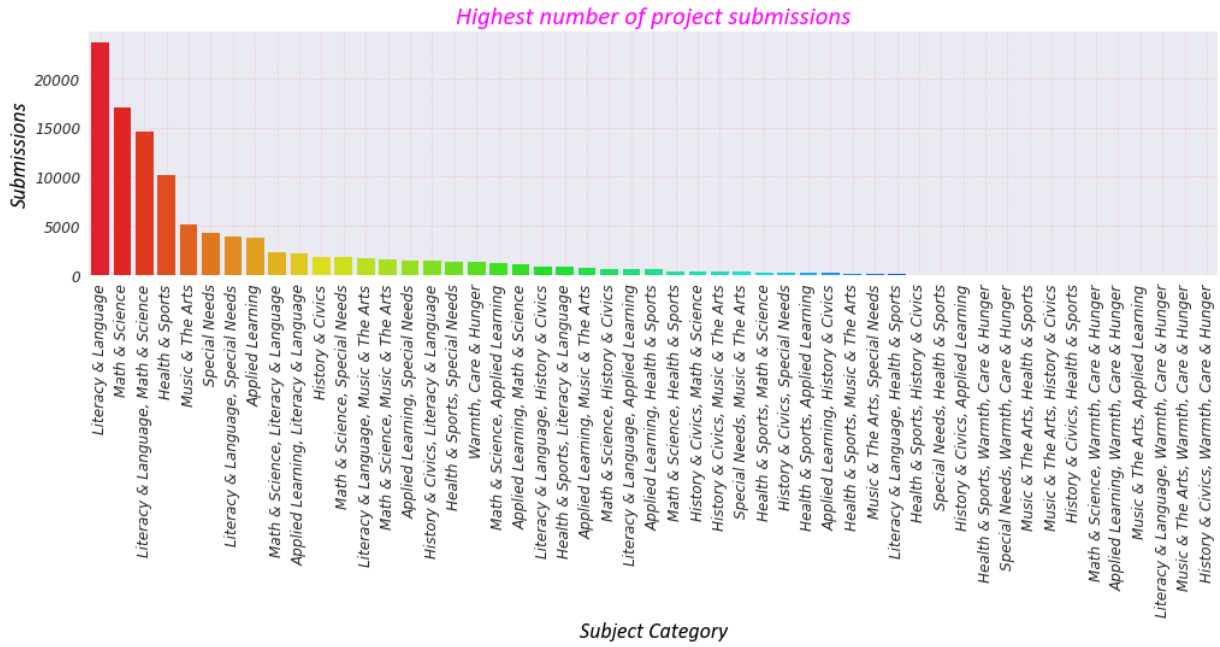



```
 *** Highest project rejection ratio ***
+------+------------------------+-------------------+-------------------+----------
+
| Seq. | project_grade_category | Reject_Submissions | Total_Submissions | Reject_%
|
+------+------------------------+-------------------+-------------------+----------
+
|  0   |      Grades 9-12       |        1780       |       10963       |  16.24
|
|  1   |       Grades 6-8       |        2665       |       16923       |  15.75
|
|  2   |     Grades PreK-2      |        6689       |       44225       |  15.12
|
|  3   |       Grades 3-5       |        5408       |       37137       |  14.56
|
+------+------------------------+-------------------+-------------------+----------
+
```

Attribute-4

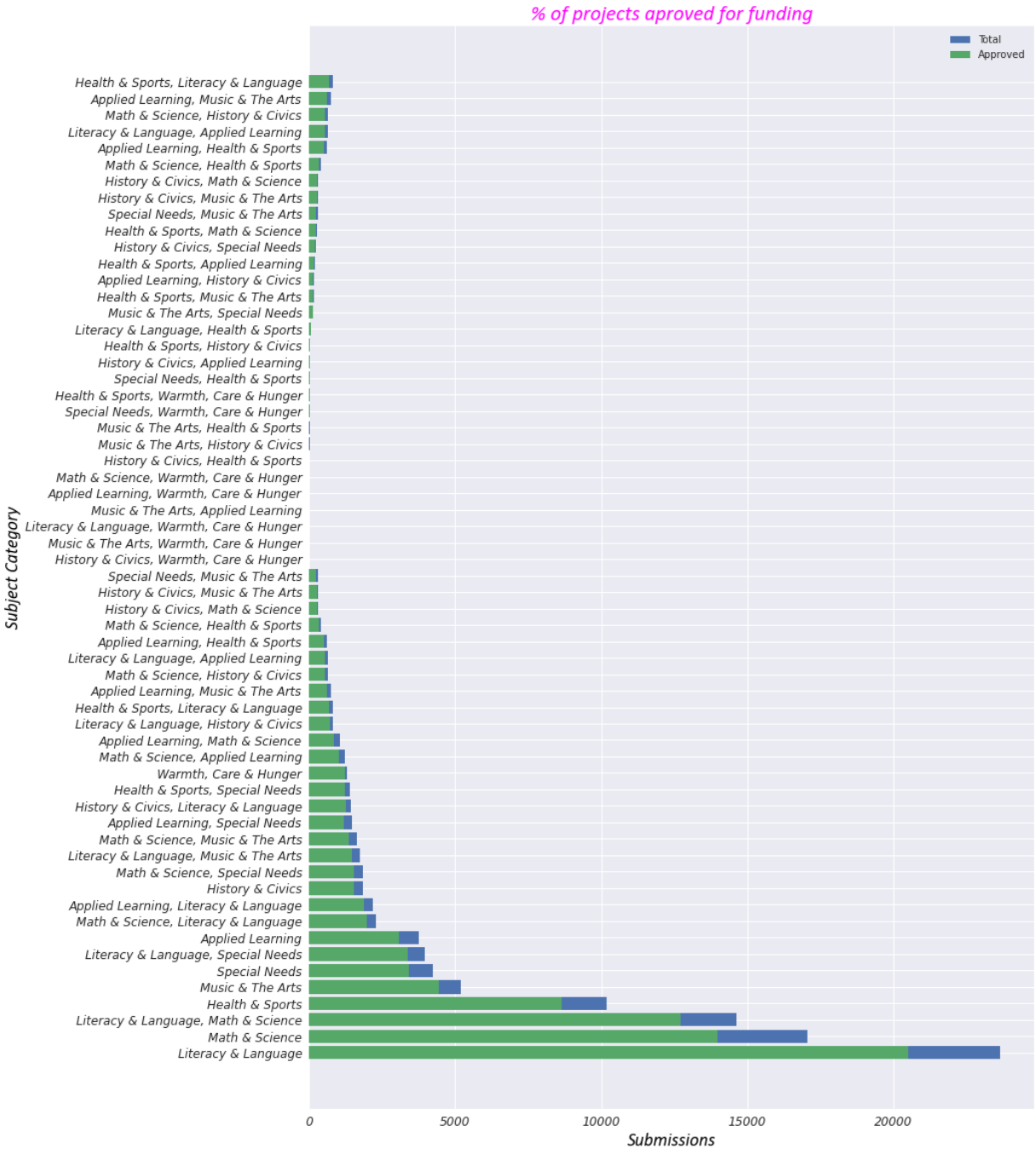
- **Project Subject Category**

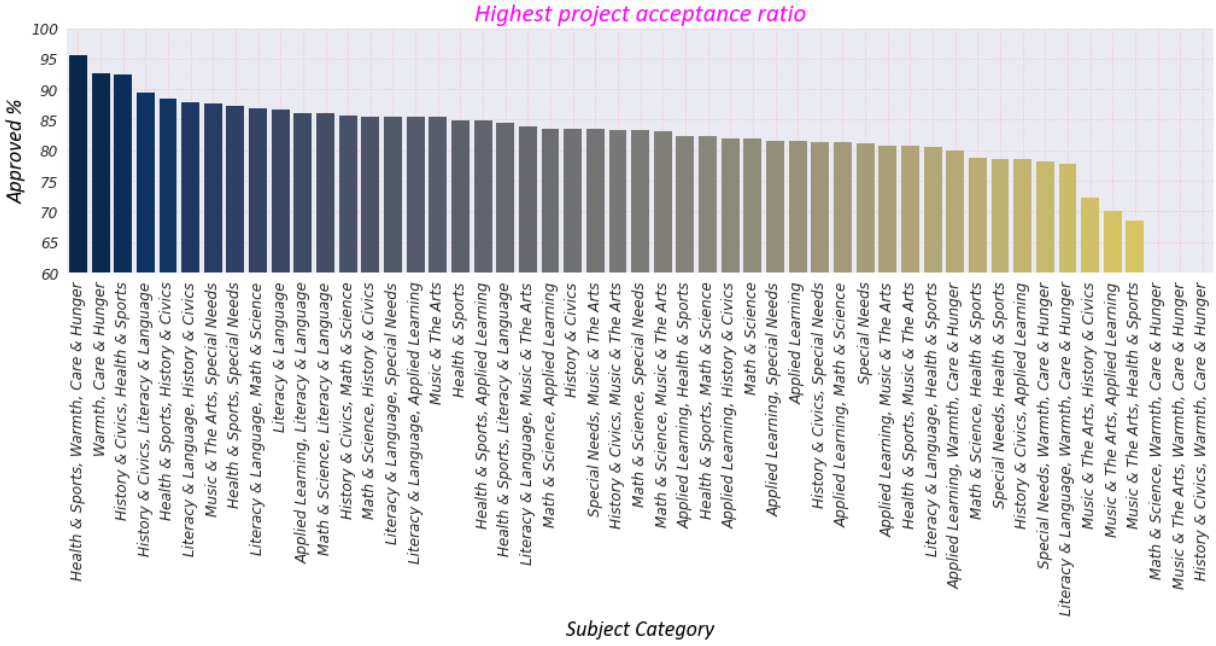
In [35]:
```
univariate_analysis(dc_train_df,'project_is_approved','project_subject_categories','
```



*** Highest number of project submissions ***

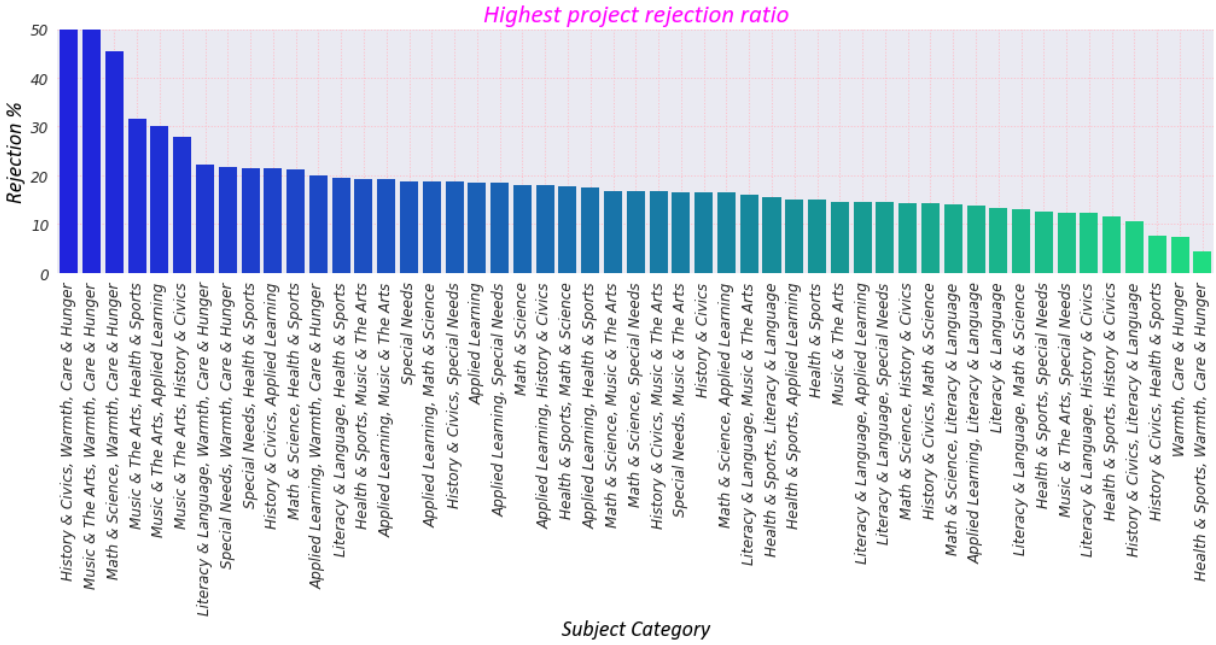
Seq.	project_subject_categories	Total_Submissions
0	Literacy & Language	23655
1	Math & Science	17072
2	Literacy & Language, Math & Science	14636
3	Health & Sports	10177
4	Music & The Arts	5180
5	Special Needs	4226
6	Literacy & Language, Special Needs	3961
7	Applied Learning	3771
8	Math & Science, Literacy & Language	2289
9	Applied Learning, Literacy & Language	2191
10	History & Civics	1851
11	Math & Science, Special Needs	1840
12	Literacy & Language, Music & The Arts	1757
13	Math & Science, Music & The Arts	1642
14	Applied Learning, Special Needs	1467
15	History & Civics, Literacy & Language	1421






```
*** Highest project acceptance ratio ***
```

Seq.	project_subject_categories	Apprv_Submissions	Total_Submissions	Apprv_%
0	Health & Sports, Warmth, Care & Hunger	22	23	95.65
1	Warmth, Care & Hunger	1212	1309	92.59
2	History & Civics, Health & Sports	12	13	92.31
3	History & Civics, Literacy & Language	1271	1421	89.44
4	Health & Sports, History & Civics	38	43	88.37
5	Literacy & Language, History & Civics	710	809	87.76
6	Music & The Arts, Special Needs	121	138	87.68
7	Health & Sports, Special Needs	1215	1391	87.35
8	Literacy & Language, Math & Science	12725	14636	86.94
9	Literacy & Language	20520	23655	86.75
10	Applied Learning, Literacy & Language	1887	2191	86.13
11	Math & Science, Literacy & Language	1968	2289	85.98
12	History & Civics, Math & Science	276	322	85.71
13	History & Civics, Math & Science	276	322	85.71
14	Math & Science, History & Civics	558	652	85.58
15	Math & Science, History & Civics	558	652	85.58



*** Highest project rejection ratio ***

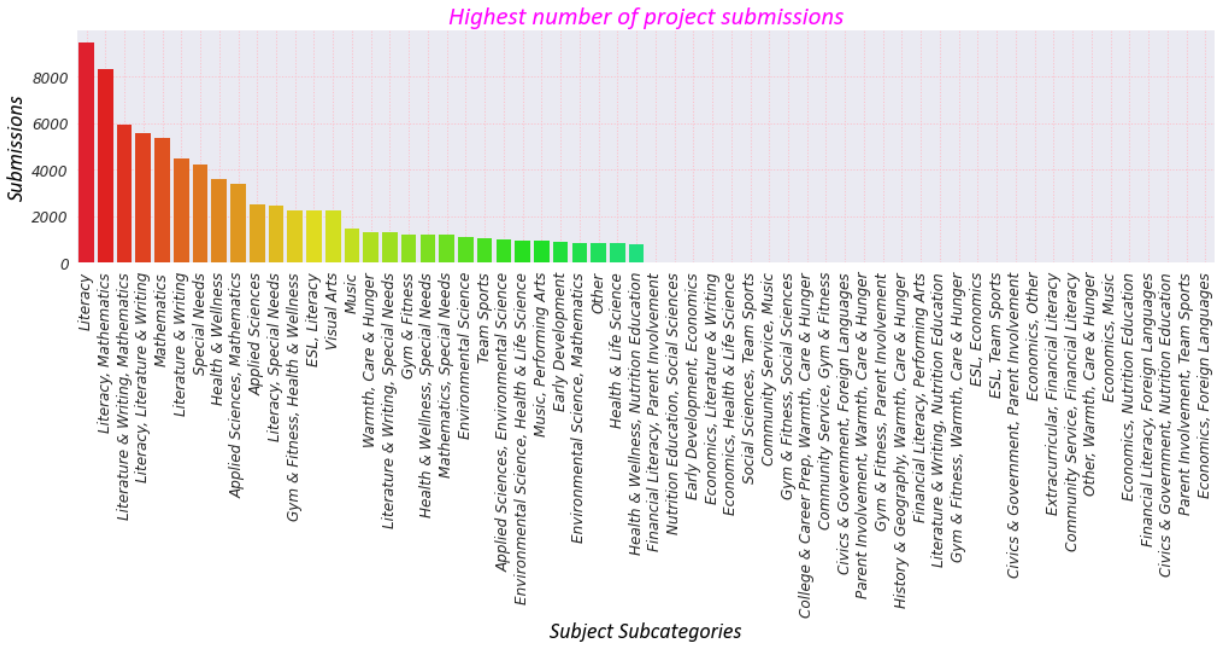
Seq.	project_subject_categories	Reject_Submissions	Total_Submissions	Reject_%

```
---------+----------+
|  0   |  History & Civics, Warmth, Care & Hunger   |         1          |          1
|  100.0   |
|  1   |  Music & The Arts, Warmth, Care & Hunger   |         1          |          2
|   50.0   |
|  2   |   Math & Science, Warmth, Care & Hunger    |         5          |          1
1        |  45.45   |
|  3   |     Music & The Arts, Health & Sports      |         6          |          1
9        |  31.58   |
|  4   |     Music & The Arts, Applied Learning     |         3          |          1
0        |   30.0   |
|  5   |     Music & The Arts, History & Civics     |         5          |          1
8        |  27.78   |
|  6   | Literacy & Language, Warmth, Care & Hunger |         2          |          9
|  22.22   |
|  7   |    Special Needs, Warmth, Care & Hunger    |         5          |          2
3        |  21.74   |
|  8   |       Special Needs, Health & Sports       |         9          |          4
2        |  21.43   |
|  9   |     History & Civics, Applied Learning     |         9          |          4
2        |  21.43   |
|  10  |      Math & Science, Health & Sports       |         88         |         41
4        |  21.26   |
|  11  |      Math & Science, Health & Sports       |         88         |         41
4        |  21.26   |
|  12  |  Applied Learning, Warmth, Care & Hunger   |         2          |          1
0        |   20.0   |
|  13  |    Literacy & Language, Health & Sports    |         14         |          7
2        |  19.44   |
|  14  |     Health & Sports, Music & The Arts      |         30         |         15
5        |  19.35   |
|  15  |     Applied Learning, Music & The Arts     |        146         |         75
8        |  19.26   |
+------+--------------------------------------------+--------------------+----------
---------+----------+
```

Attribute-5

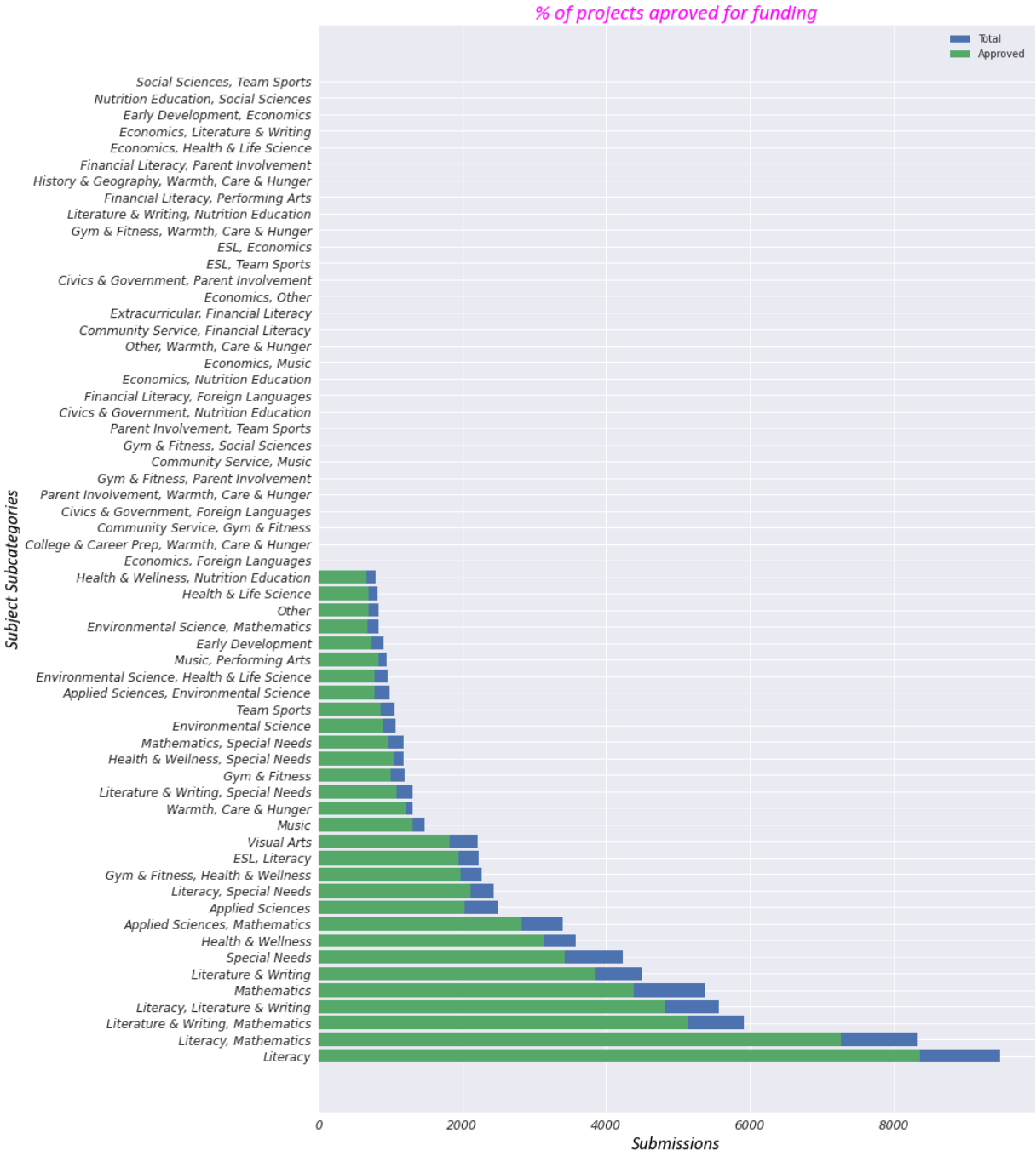
- **Project Subject Sub-Category**

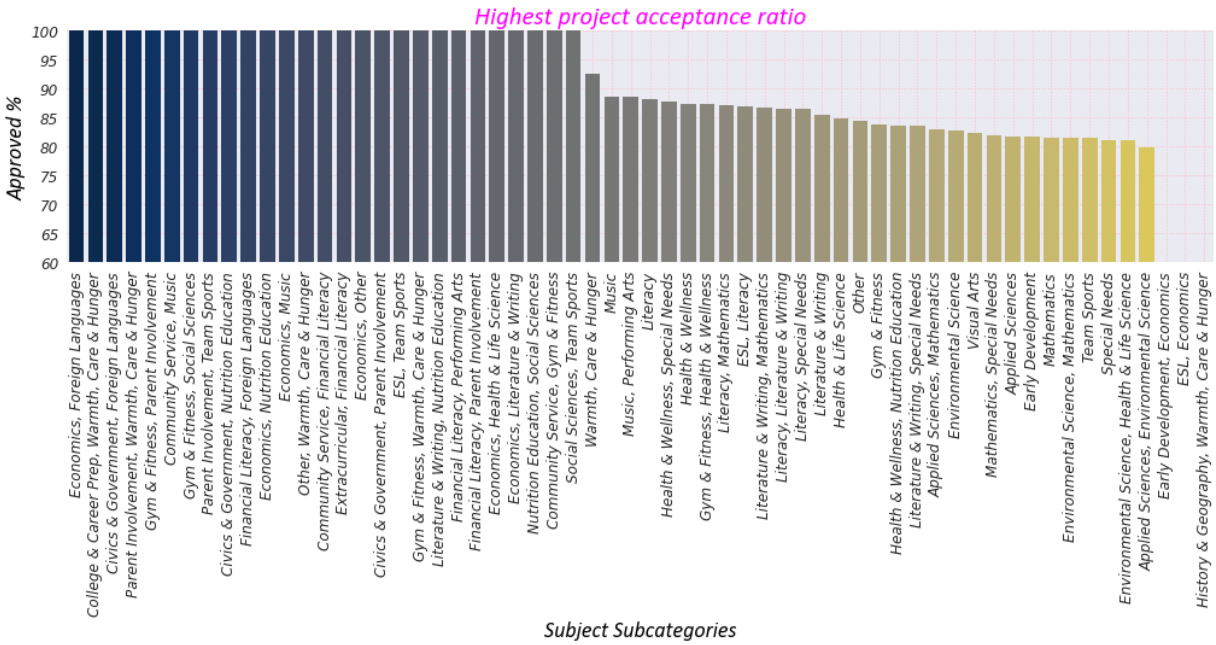
In [36]:
```
univariate_analysis(dc_train_df,'project_is_approved','project_subject_subcategories
```




```
 *** Highest number of project submissions ***
+------+-----------------------------------+------------------+
| Seq. |   project_subject_subcategories   | Total_Submissions |
```

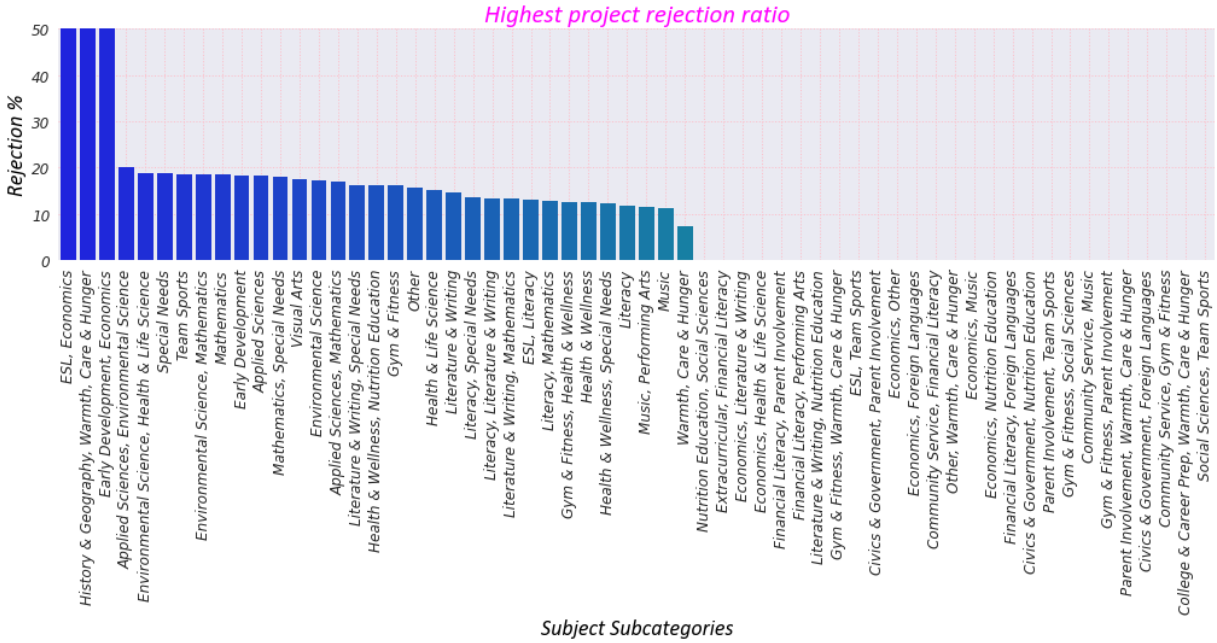
0	Literacy	9486
1	Literacy, Mathematics	8325
2	Literature & Writing, Mathematics	5923
3	Literacy, Literature & Writing	5571
4	Mathematics	5379
5	Literature & Writing	4501
6	Special Needs	4226
7	Health & Wellness	3583
8	Applied Sciences, Mathematics	3399
9	Applied Sciences	2492
10	Literacy, Special Needs	2440
11	Gym & Fitness, Health & Wellness	2264
12	ESL, Literacy	2234
13	Visual Arts	2217
14	Music	1472
15	Warmth, Care & Hunger	1309





*** Highest project acceptance ratio ***

Seq.	project_subject_subcategories	Apprv_Submissions	Total_Submissions	Apprv_%
0	Economics, Foreign Languages	1	1	100.0
1	College & Career Prep, Warmth, Care & Hunger	1	1	100.0
2	Civics & Government, Foreign Languages	1	1	100.0
3	Parent Involvement, Warmth, Care & Hunger	1	1	100.0
4	Gym & Fitness, Parent Involvement	1	1	100.0
5	Community Service, Music	1	1	100.0
6	Gym & Fitness, Social Sciences	1	1	100.0
7	Parent Involvement, Team Sports	1	1	100.0
8	Civics & Government, Nutrition Education	1	1	100.0
9	Financial Literacy, Foreign Languages	1	1	100.0
10	Economics, Nutrition Education	1	1	100.0
11	Economics, Music	1	1	100.0
12	Other, Warmth, Care & Hunger	1	1	100.0
13	Community Service, Financial Literacy	1	1	100.0
14	Extracurricular, Financial Literacy	1	1	100.0
15	Economics, Other	1	1	100.0



*** Highest project rejection ratio ***

Seq.	project_subject_subcategories	Reject_Submissions	Total_Submissions	Reject_%
0	ESL, Economics	1	1	100.0
1	History & Geography, Warmth, Care & Hunger	1	1	100.0
2	Early Development, Economics	1	2	50.0
3	Applied Sciences, Environmental Science	199	984	20.22
4	Environmental Science, Health & Life Science	182	964	18.88
5	Special Needs	795	4226	18.81
6	Team Sports	197	1061	18.57
7	Environmental Science, Mathematics	155	838	18.5
8	Mathematics	994	5379	18.48
9	Early Development	165	905	18.23
10	Applied Sciences	454	2492	18.22
11	Mathematics, Special Needs	215	1187	18.11
12	Visual Arts	390	2217	17.59
13	Environmental Science	185	1079	17.15
14	Applied Sciences, Mathematics	575	3399	16.92
15	Literature & Writing, Special Needs	213	1306	16.31

Attribute-6

- **Teacher's number of previously posted projects**

In [37]:
```
teacher_prev_projs = pd.DataFrame(dc_train_df.groupby(['project_is_approved','teache
                                  count()).reset_index().rename(columns={'id':'Submi
```

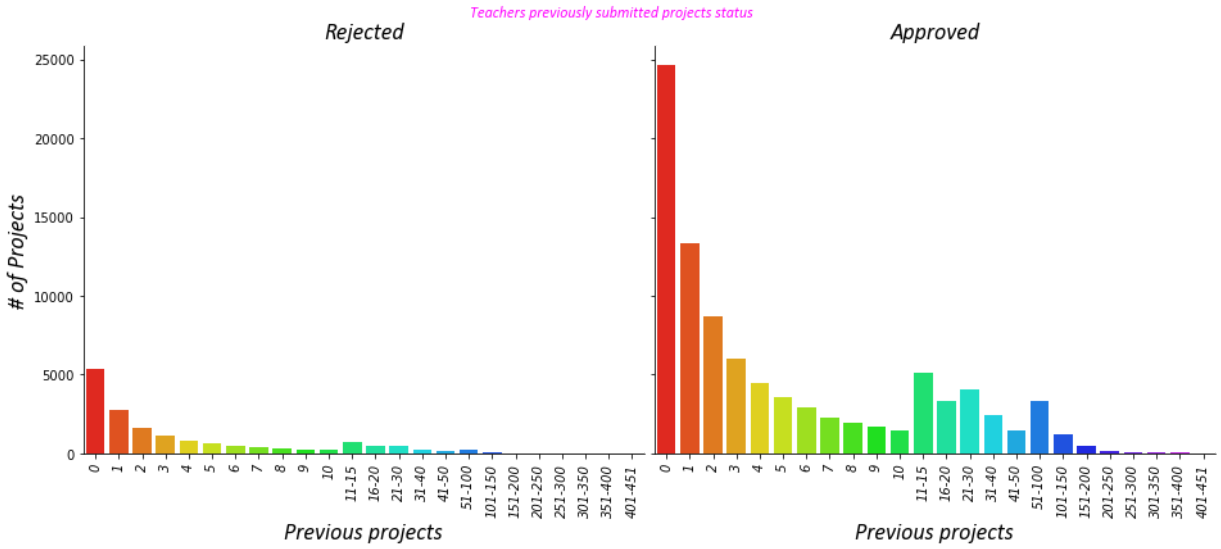
In [38]:
```
def feat_buckets(val):
    """
    Description: This function is created for segregating the features values in buc

    Input: It accepts only one parameter:
        1. `val`: int
            Feature value used for allocating its bucket.

    Return: Bucket belongs to feature value.
    """
    if val ==0:
        return '0'
    elif val ==1:
        return '1'
    elif val ==2:
        return '2'
    elif val ==3:
        return '3'
    elif val ==4:
        return '4'
    elif val ==5:
        return '5'
    elif val ==6:
        return '6'
    elif val ==7:
        return '7'
    elif val ==8:
        return '8'
    elif val ==9:
        return '9'
    elif val ==10:
        return '10'
    elif val >=11 and val <=15:
        return '11-15'
    elif val >=16 and val <=20:
        return '16-20'
    elif val >=21 and val <=30:
        return '21-30'
    elif val >=31 and val <=40:
        return '31-40'
    elif val >=41 and val <=50:
        return '41-50'
    elif val >=51 and val <=100:
        return '51-100'
    elif val >=101 and val <=150:
        return '101-150'
    elif val >=151 and val <=200:
        return '151-200'
    elif val >=201 and val <=250:
        return '201-250'
    elif val >=251 and val <=300:
        return '251-300'
    elif val >=301 and val <=350:
        return '301-350'
    elif val >=351 and val <=400:
        return '351-400'
    elif val >=401 and val <=451:
        return '401-451'
```

In [39]:
```
teacher_prev_projs['prev_proj_buckets'] = teacher_prev_projs['teacher_number_of_prev
                                          apply(lambda prev_proj: feat_buckets
```

In [40]:
```
mul_index = pd.MultiIndex.from_product([teacher_prev_projs['project_is_approved'].un
                                        names=['project_is_approved','prev_proj_bucke
```

In [41]:
```
teacher_prev_projs_bkts = teacher_prev_projs.groupby(['project_is_approved','prev_pr
                                                      reindex(mul_index, fill_value=0)
```

In [42]:
```
prev_projs = sns.catplot(data=teacher_prev_projs_bkts,x='prev_proj_buckets',y='Submi
                         kind='bar',orient='v',palette='gist_rainbow',aspect=1.1,hei
axes = prev_projs.axes.flatten()
axes[0].set_title("Rejected",fontdict=lbl_dict)
axes[0].set_xlabel("Previous projects",fontdict=lbl_dict)
axes[0].set_ylabel("# of Projects",fontdict=lbl_dict)
axes[1].set_title("Approved",fontdict=lbl_dict)
axes[1].set_xlabel("Previous projects",fontdict=lbl_dict)
axes[1].set_ylabel("",fontdict=lbl_dict)
prev_projs.fig.suptitle("Teachers previously submitted projects status",fontdict=ttl
prev_projs.set_xticklabels(rotation=90,size=10,style='oblique')
plt.tight_layout(pad=0.5,h_pad=0.3)
plt.show()
```



The above plot gives us the comparison b/w approved and rejected projects based on number of projects previously submitted by teachers. Here, we need to jot down few important points:

- **Teachers with no previous submissions holds the highest approval and rejection submissions.**
- **This graph tells us that if the teacher has submitted the project previously then the approval is quite high.**

In [43]:
```
no_prev_projs = dc_train_df[dc_train_df['teacher_number_of_previously_posted_project
one_more_prev_projs = dc_train_df[dc_train_df['teacher_number_of_previously_posted_p

no_prev_projs, one_more_prev_projs
```

Out[43]: (30014, 79234)

In [44]:
```
accepted_projs = dc_train_df['project_is_approved'].value_counts()[1]
rejected_projs = dc_train_df['project_is_approved'].value_counts()[0]

accepted_projs, rejected_projs
```

Out[44]: (92706, 16542)

In [45]:
```
prev_projs_yes = pd.DataFrame(dc_train_df[dc_train_df['teacher_number_of_previously_
                              .groupby(['project_is_approved'])['id'].count()).reset
prev_projs_yes['subms_with_prev_projs_yes'] = [one_more_prev_projs,one_more_prev_pro
prev_projs_yes['overall_proj_subms'] = [rejected_projs,accepted_projs]
prev_projs_yes['perc_subms'] = (prev_projs_yes['proj_subms']/prev_projs_yes['subms_w
prev_projs_yes
```

Out[45]:

	project_is_approved	proj_subms	subms_with_prev_projs_yes	overall_proj_subms	perc_subms
0	0	11180	79234	16542	14.110104
1	1	68054	79234	92706	85.889896

The above table tells us that approx. 86% of proposals were approved when the previous submitted projects were 1 or more.

In [46]:
```
prev_projs_no = pd.DataFrame(dc_train_df[dc_train_df['teacher_number_of_previously_p
                              .groupby(['project_is_approved'])['id'].count()).reset
prev_projs_no['subms_with_no_prev_projs'] = [no_prev_projs,no_prev_projs]
prev_projs_no['overall_proj_subms'] = [rejected_projs,accepted_projs]
prev_projs_no['perc_subms'] = (prev_projs_no['proj_subms']/prev_projs_no['subms_with
prev_projs_no
```

Out[46]:

	project_is_approved	proj_subms	subms_with_no_prev_projs	overall_proj_subms	perc_subms
0	0	5362	30014	16542	17.864996
1	1	24652	30014	92706	82.135004

The above table tells us that approx. 82% of proposals were approved when there were no previous submitted projects.

Thus, by seeing the above ratios we can say that the previous submitted projects status doesn't really ensures that the current submission will surely be accepted.

Digging more at the teachers level

Unique teachers in dataset

In [47]:
```
unique_tids = dc_train_df['teacher_id'].nunique()
print("Total number of unique teachers in the dataset --> {}".format(format(unique_t
```

```
Total number of unique teachers in the dataset --> 72,168
```

Teacher IDs with zero or more previous submissions

In [48]:
```
teacher_details = dc_train_df.groupby(['teacher_id'])['teacher_number_of_previously_
.sort_values(by='max',ascending=True)

tids_with_one_or_more_prev_proj = teacher_details[(teacher_details['min']>=0) & (tea
teachers_with_one_or_more_prev_proj = len(tids_with_one_or_more_prev_proj)

tids_with_no_prev_proj = teacher_details[(teacher_details['min']==0) & (teacher_deta
teachers_with_no_prev_proj = len(tids_with_no_prev_proj)

print("Number of teachers with 1 or more previously submitted projects --> {}".forma
print("Number of teachers with 0 or no previously submitted projects --> {}".format(
print("Total Numbers of teachers with previously zero or more submissions --> {}"\
      .format(format((teachers_with_one_or_more_prev_proj+teachers_with_no_prev_proj
```

```
Number of teachers with 1 or more previously submitted projects --> 46,827
Number of teachers with 0 or no previously submitted projects --> 25,341
Total Numbers of teachers with previously zero or more submissions --> 72,168
```

Records in dataset of teachers who previously submitted 1 or more projects

In [49]:
```
recs_with_one_more_prev_proj = dc_train_df[dc_train_df['teacher_id'].isin(tids_with_
print("Number of records in dataset of teachers with 1 or more previously submitted
      .format(format(recs_with_one_more_prev_proj,',d')))
```

```
Number of records in dataset of teachers with 1 or more previously submitted project
s --> 83,502
```

In [50]:
```
tids_yes = dc_train_df[dc_train_df['teacher_id'].isin(tids_with_one_or_more_prev_pro
.groupby(['project_is_approved'])['id'].count().reset_index().rename(columns={'id':'

tids_yes['Ratio_%'] = (tids_yes['Submissions']/recs_with_one_more_prev_proj)*100
tids_yes
```

Out[50]:

	project_is_approved	Submissions	Ratio_%
0	0	11802	14.133793
1	1	71700	85.866207

Records in dataset of teachers who previously had no submissions

In [51]:
```
recs_with_no_prev_proj = dc_train_df[dc_train_df['teacher_id'].isin(tids_with_no_pre
print("Number of records in dataset of teachers with no previoussubmissions --> {}".
```

```
Number of records in dataset of teachers with no previoussubmissions --> 25,746
```

In [52]:
```
tids_no = dc_train_df[dc_train_df['teacher_id'].isin(tids_with_no_prev_proj)]\
.groupby(['project_is_approved'])['id'].count().reset_index().rename(columns={'id':'

tids_no['Ratio_%'] = (tids_no['Submissions']/recs_with_no_prev_proj)*100
tids_no
```

Out[52]:

	project_is_approved	Submissions	Ratio_%
0	0	4740	18.410627
1	1	21006	81.589373

So, the above ratios telling us that we have a quite similar % of acceptance for teachers who either had previous submissions or not.

- **Approx. 86% of proposals accepted from teachers who previously submitted one more projects. And, approx. 82% of proposals were accepted from teachers who didn't had any past submissions.**

Attribute-7

- `Project Resource Summary`

In [53]:
```
indxs = np.random.randint(0,100000,14)
for idx in indxs:
    print(idx,'::',dc_train_df['project_resource_summary'].iloc[idx],'::',dc_train_d

### Odd ones out : 46654,60959,12
print(12,'::',dc_train_df['project_resource_summary'].iloc[12],'::',dc_train_df['pro
print(46654,'::',dc_train_df['project_resource_summary'].iloc[46654],'::',dc_train_d
print(60959,'::',dc_train_df['project_resource_summary'].iloc[60959],'::',dc_train_d
```

```
85876 :: My students need a computer to be able to integrate technology in their dai
ly learning process, the Lenovo 300-22ACL 21.5\" All-In-One Desktop Computer will be
a great acquisition for them as we have only two. :: 1
```

```
78340 :: My students need more weight for our weight room project in 2017 :: 1
30960 :: My students need indoor balls, foam frisbees, exercise ball seats and playg
round equipment to et fit at school. :: 1
21152 :: My students need an iPad mini 2 for our classroom. We currently have two iP
ads and my students are so eager to increase their learning skills when using this t
echnology! :: 1
73315 :: My students need a projector and screen to give them access to better resou
rces and to help display their work in the classroom. :: 1
72441 :: My students need tablets to bring our classroom into the 21st century so th
ey can practice their STEM skills on real world activities and create a College Cult
ure Classroom Project! :: 1
92588 :: My students need two iPad tablets with 2 shockproof case cover. :: 1
59978 :: My students need a 3doodler Start Full Education Bundle containing pens, do
odle-pads, nozzles, jet-packs, and plastic strands. :: 1
24575 :: My students need safety goggles to protect their eyes while doing labs. ::
1
38824 :: My students need a variety of children's books to listen to as read alouds,
select for their book boxes or share with friends. :: 1
55235 :: My students need farm animals, forest animals, ocean animals, wildlife anim
als, books about animals, a bookshelf and animal puzzles to learn about wildlife and
their habitats. :: 1
70444 :: My students need help with being organized.  These storage cubes will free
up valuable desk space so they have what they need at their fingertips. :: 1
3698 :: My students need a set of new books to read and share. :: 1
95841 :: My students need art materials, classroom organizers like the store-it-all
rotating caddies to organize their writing materials. My goal is to encourage the lo
ve of science and math careers and the materials will support it. :: 0
12 :: My students need 3D and 4D life science activity kits so they can get the full
effect of the lessons and work in their groups and store their models in sturdy bin
s. :: 0
46654 :: My students need 20 Lenovo N22 Chromebook computers to aid them in their sc
ientific learning. :: 0
60959 :: My students need 25 stability balls and 4 wobble cushions to help focus lea
rning during seated tasks. :: 0
```

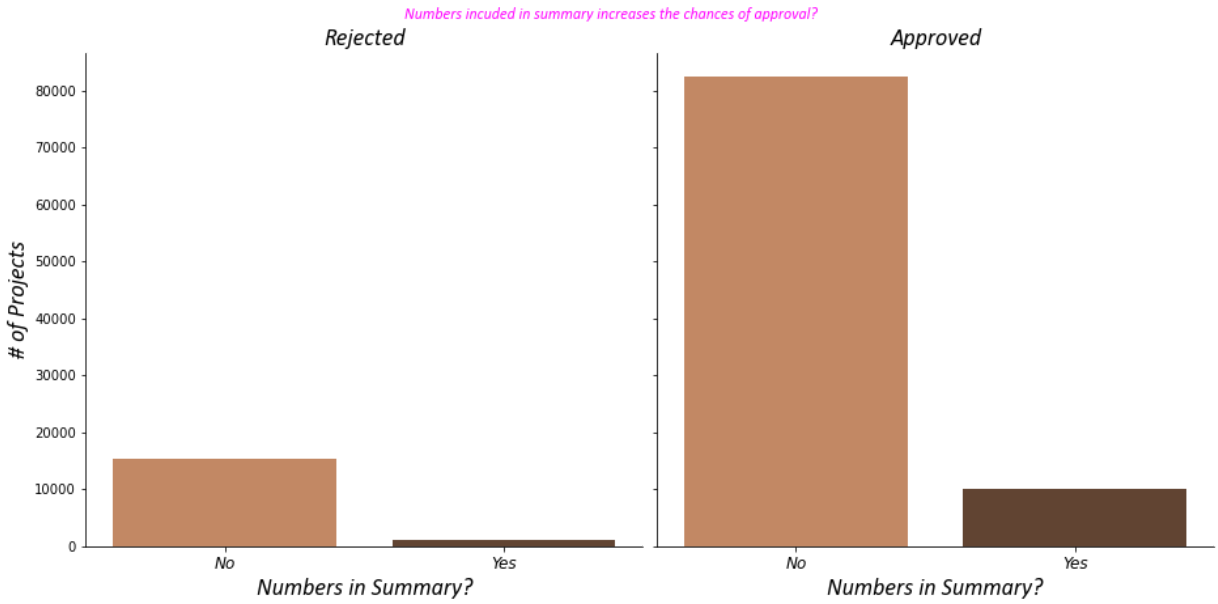
In [54]:

```python
def check_nums_in_summary(text):
    """
    Description: This function is created for return the boolean value based on exis

    Inputs: It accepts below parameters:
        1. `text`: str
            Text in which numbers existence needs to be found.

    Return: boolean value(1 if numbers exists else 0)

    `NOTE`: I'm finding the pure numbers in the text. Refer below examples:

    "need 25 stability balls" --> 25 is pure number here --> thus, function will ret
    "need 4 wobble cushions" --> 4 is pure number here --> thus, function will retur
    "need 3D and 4D life science activity kits" --> no pure number here --> thus, fu
    "need 20 Lenovo N22 Chromebook computers" --> 20 is only a pure number here -->
    """
    contain_nums = []
    for word in text.split(" "):
        res = re.findall('[0-9]',word)
        if res != []:
            if len(res) == len(word):
                'Actual/Pure numbers are found'
                contain_nums.append(word)
            else:
                'No numbers'

    if len(contain_nums) != 0:
        return 1
    else:
        return 0
```

In [55]:
```
dc_train_df['nums_in_res_summary'] = dc_train_df['project_resource_summary'].apply(l
```

In [56]:
```
res_sumry_nums = dc_train_df.groupby(['project_is_approved','nums_in_res_summary'])[
                 .rename(columns={'id':'submissions'})
```

In [57]:
```
proj_smry = sns.catplot(data=res_sumry_nums,x='nums_in_res_summary',y='submissions',
                        kind='bar',orient='v',palette='copper_r',aspect=1,height=6.
axes = proj_smry.axes.flatten()
axes[0].set_title("Rejected",fontdict=lbl_dict)
axes[0].set_xlabel("Numbers in Summary?",fontdict=lbl_dict)
axes[0].set_ylabel("# of Projects",fontdict=lbl_dict)
axes[1].set_title("Approved",fontdict=lbl_dict)
axes[1].set_xlabel("Numbers in Summary?",fontdict=lbl_dict)
axes[1].set_ylabel("",fontdict=lbl_dict)
proj_smry.fig.suptitle("Numbers incuded in summary increases the chances of approval
proj_smry.set_xticklabels(['No','Yes'],rotation=0,size=12,style='oblique')
plt.tight_layout(pad=0.7,h_pad=0.3)
plt.show()
```



The above graph definitely suggesting us that quite a lot of projects with numerics in their summaries approved significantly. But, at the same time it is also giving us information that good number of projects with no numerical digits in their summaries were approved.

In [58]:
```
res_sumry_nums['tot_subms'] = res_sumry_nums['nums_in_res_summary']\
.apply(lambda val: dc_train_df['nums_in_res_summary'].value_counts()[0] if val ==0 e

res_sumry_nums['Ratio_%'] = (res_sumry_nums['submissions']*100.0)/res_sumry_nums['to

res_sumry_nums
```

Out[58]:

	project_is_approved	nums_in_res_summary	submissions	tot_subms	Ratio_%
0	0	0	15449	98011	15.762516
1	0	1	1093	11237	9.726795
2	1	0	82562	98011	84.237484
3	1	1	10144	11237	90.273205

The above table clears all the doubts approx. 85% of proposals are approved with no digits in the summary whereas approx. 90% of projects were approved with numerics in their summaries.

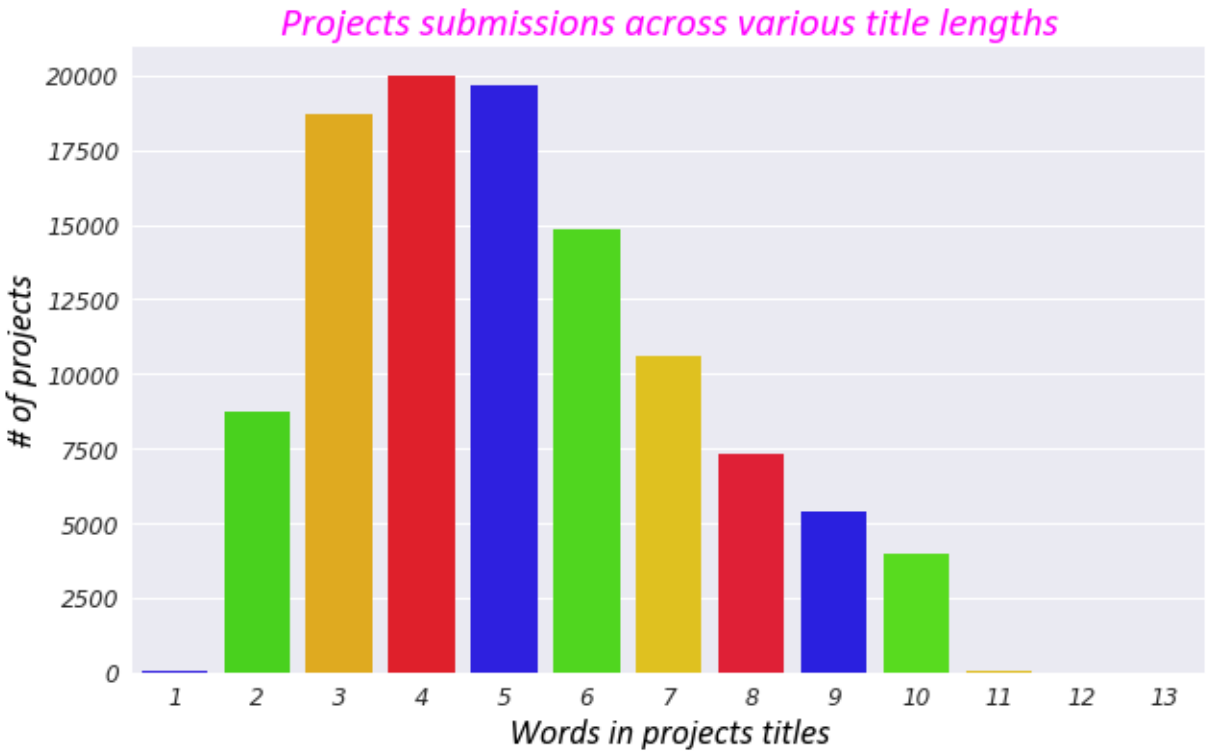
And, if the project summary has digits then the rejection % is less than 10%. However, the rejection is ratio is also quite less(approx. 16%) when there are no numerical alphabets in the summary.

Attribute-8

- **Project Title**

In [59]:
```
proj_ttl_len = pd.DataFrame(dc_train_df['project_title'].str.split().apply(lambda ro
                    .reset_index().rename(columns={'index':'title_lengths','project_
                    .sort_values(by='num_submissions',ascending=False)
```

In [60]:
```
with plt.style.context('seaborn'):
    plt.figure(figsize=(10,6))
    sns.barplot(x='title_lengths',y='num_submissions',data=proj_ttl_len,palette='pri
    plt.xlabel('Words in projects titles',fontdict=lbl_dict)
    plt.ylabel('# of projects',fontdict=lbl_dict)
    plt.title('Projects submissions across various title lengths',fontdict=ttl_dict)
    plt.xticks(size=12,style='oblique')
    plt.yticks(size=12,style='oblique')
```



In [61]:
```
proj_ttl_len.head(5)
```

Out[61]:

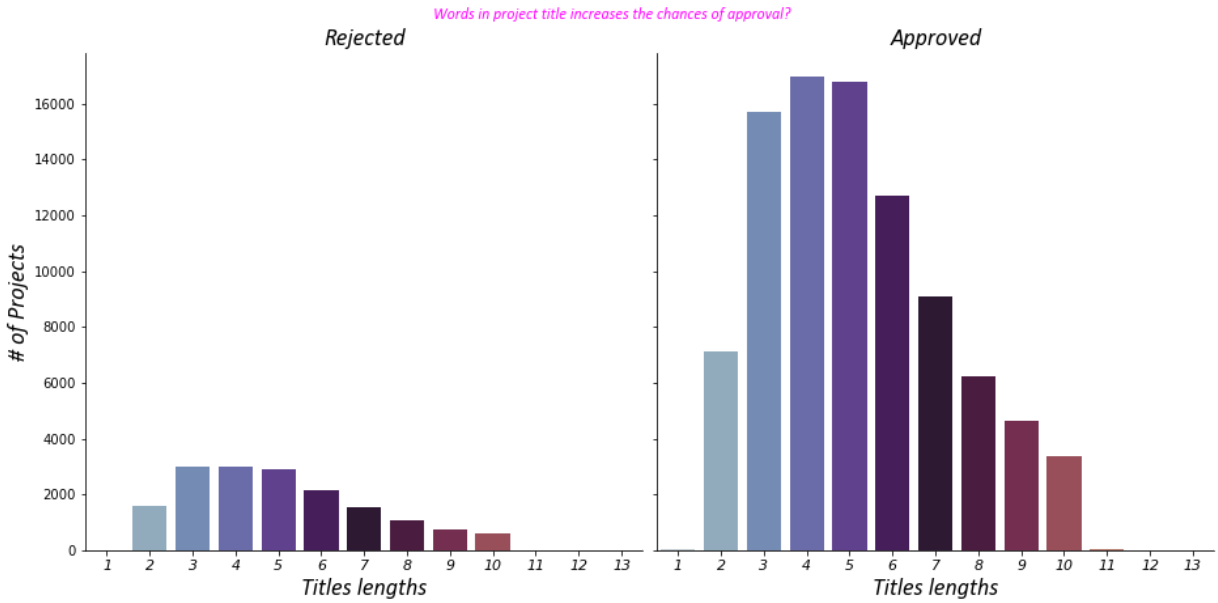
	title_lengths	num_submissions
0	4	19979
1	5	19677
2	3	18691
3	6	14824
4	7	10631

Here, we found that the most popular title lengths are with number of words 3,4,5,6 and 7.

In [62]:
```
dc_train_df['proj_ttl_wrd_len'] = dc_train_df['project_title'].str.split().apply(lam
```

In [63]:
```
proj_ttl_apprv_status = pd.DataFrame(dc_train_df.groupby(['project_is_approved','pro
                reset_index().rename(columns={'id':'submissions'})
```

In [64]:
```
projs_ttls = sns.catplot(data=proj_ttl_apprv_status,x='proj_ttl_wrd_len',y='submissi
                         kind='bar',orient='v',palette='twilight',aspect=1,height=6.
axes = projs_ttls.axes.flatten()
axes[0].set_title("Rejected",fontdict=lbl_dict)
axes[0].set_xlabel("Titles lengths",fontdict=lbl_dict)
axes[0].set_ylabel("# of Projects",fontdict=lbl_dict)
axes[1].set_title("Approved",fontdict=lbl_dict)
axes[1].set_xlabel("Titles lengths",fontdict=lbl_dict)
axes[1].set_ylabel("",fontdict=lbl_dict)
projs_ttls.fig.suptitle("Words in project title increases the chances of approval?",
projs_ttls.set_xticklabels(rotation=0,size=11,style='oblique')
plt.tight_layout(pad=0.7,h_pad=0.3)
plt.show()
```



In [65]:
```
proj_ttl_apprv_status['tot_subms']=proj_ttl_apprv_status['proj_ttl_wrd_len']\
.apply(lambda row: proj_ttl_len[proj_ttl_len['title_lengths']==row]['num_submissions

proj_ttl_apprv_status['Ratio_%']=np.round((proj_ttl_apprv_status['submissions']*100.

proj_ttl_apprv_status
```

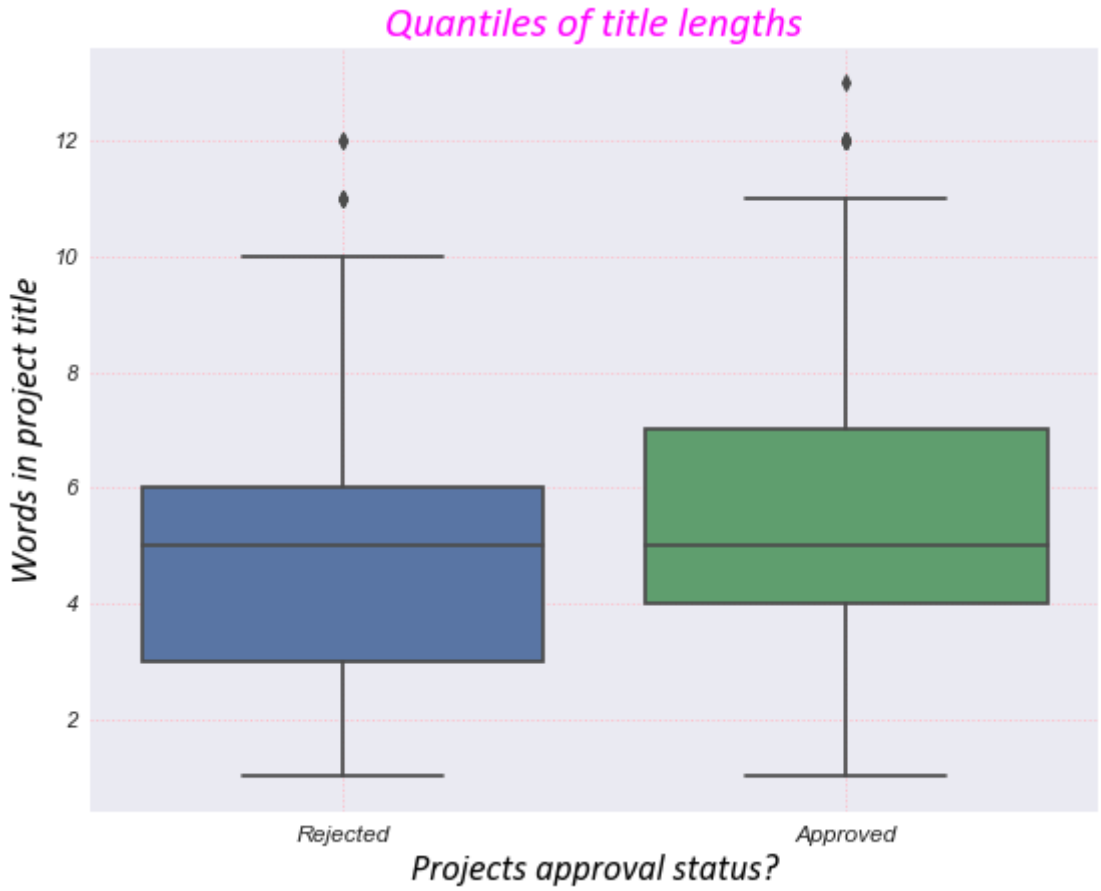
Out[65]:

	project_is_approved	proj_ttl_wrd_len	submissions	tot_subms	Ratio_%
0	0	1	4	31	12.90
1	0	2	1588	8733	18.18
2	0	3	2978	18691	15.93
3	0	4	2998	19979	15.01
4	0	5	2893	19677	14.70
5	0	6	2138	14824	14.42
6	0	7	1538	10631	14.47
7	0	8	1058	7289	14.52
8	0	9	750	5383	13.93

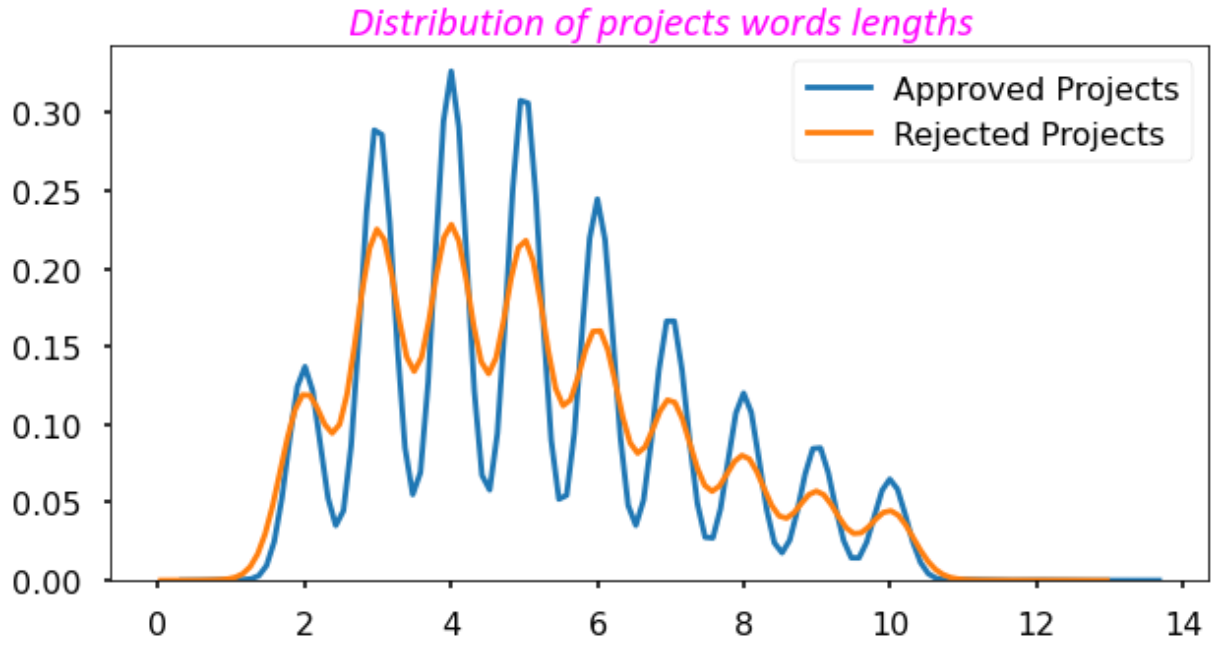
	project_is_approved	proj_ttl_wrd_len	submissions	tot_subms	Ratio_%
9	0	10	592	3968	14.92
10	0	11	3	30	10.00
11	0	12	2	11	18.18
12	1	1	27	31	87.10
13	1	2	7145	8733	81.82
14	1	3	15713	18691	84.07
15	1	4	16981	19979	84.99
16	1	5	16784	19677	85.30
17	1	6	12686	14824	85.58
18	1	7	9093	10631	85.53
19	1	8	6231	7289	85.48
20	1	9	4633	5383	86.07
21	1	10	3376	3968	85.08
22	1	11	27	30	90.00
23	1	12	9	11	81.82
24	1	13	1	1	100.00

The above table tells us that we have almost similar percentage of acceptance accross all the title lengths.

In [66]:
```
with plt.style.context('seaborn'):
    plt.figure(figsize=(9,7))
    sns.boxplot(data=[dc_train_df[dc_train_df['project_is_approved']==0]['proj_ttl_w
                      dc_train_df[dc_train_df['project_is_approved']==1]['proj_ttl_w
    plt.xlabel("Projects approval status?",fontdict=lbl_dict)
    plt.xticks([0,1],('Rejected','Approved'),style='oblique',size=12)
    plt.ylabel('Words in project title',fontdict=lbl_dict)
    plt.yticks(style='oblique',size=11)
    plt.title('Quantiles of title lengths',fontdict=ttl_dict)
    plt.grid(which='major',linestyle=':',color='pink')
    plt.show()
```



In [67]:
```
with plt.style.context('seaborn-poster'):
    plt.figure(figsize=(10,5))
    sns.distplot(dc_train_df[dc_train_df['project_is_approved']==1]['proj_ttl_wrd_le
    sns.distplot(dc_train_df[dc_train_df['project_is_approved']==0]['proj_ttl_wrd_le
    plt.title("Distribution of projects words lengths",fontdict=ttl_dict)
    plt.show()
```



Attribute-9

- **Project Essay's**

In [68]:
```
dc_train_df['essays'] = pd.DataFrame(dc_train_df["project_essay_1"].map(str) +\
                                     dc_train_df["project_essay_2"].map(str) +\
```

```
                                    dc_train_df["project_essay_3"].map(str) +\
                                    dc_train_df["project_essay_4"].map(str))
```

In [69]:
```
dc_train_df['essays'].iloc[0]
```

Out[69]:
```
'My students are English learners that are working on English as their second or thi
rd languages. We are a melting pot of refugees, immigrants, and native-born American
s bringing the gift of language to our school. \\r\\n\\r\\n We have over 24 language
s represented in our English Learner program with students at every level of master
y.  We also have over 40 countries represented with the families within our school.
Each student brings a wealth of knowledge and experiences to us that open our eyes t
o new cultures, beliefs, and respect.\\"The limits of your language are the limits o
f your world.\\"-Ludwig Wittgenstein  Our English learner\'s have a strong support s
ystem at home that begs for more resources.  Many times our parents are learning to
read and speak English along side of their children.  Sometimes this creates barrier
s for parents to be able to help their child learn phonetics, letter recognition, an
d other reading skills.\\r\\n\\r\\nBy providing these dvd\'s and players, students a
re able to continue their mastery of the English language even if no one at home is
able to assist.  All families with students within the Level 1 proficiency status, w
ill be a offered to be a part of this program.  These educational videos will be spe
cially chosen by the English Learner Teacher and will be sent home regularly to watc
h.  The videos are to help the child develop early reading skills.\\r\\n\\r\\nParent
s that do not have access to a dvd player will have the opportunity to check out a d
vd player to use for the year.  The plan is to use these videos and educational dvd
\'s for the years to come for other EL students.\\r\\nnannan'
```

In [70]:
```
proj_essay_len = pd.DataFrame(dc_train_df['essays'].str.split().apply(lambda row: le
                    .reset_index().rename(columns={'index':'essays_lengths','essays'
                    .sort_values(by='num_submissions',ascending=False)
```

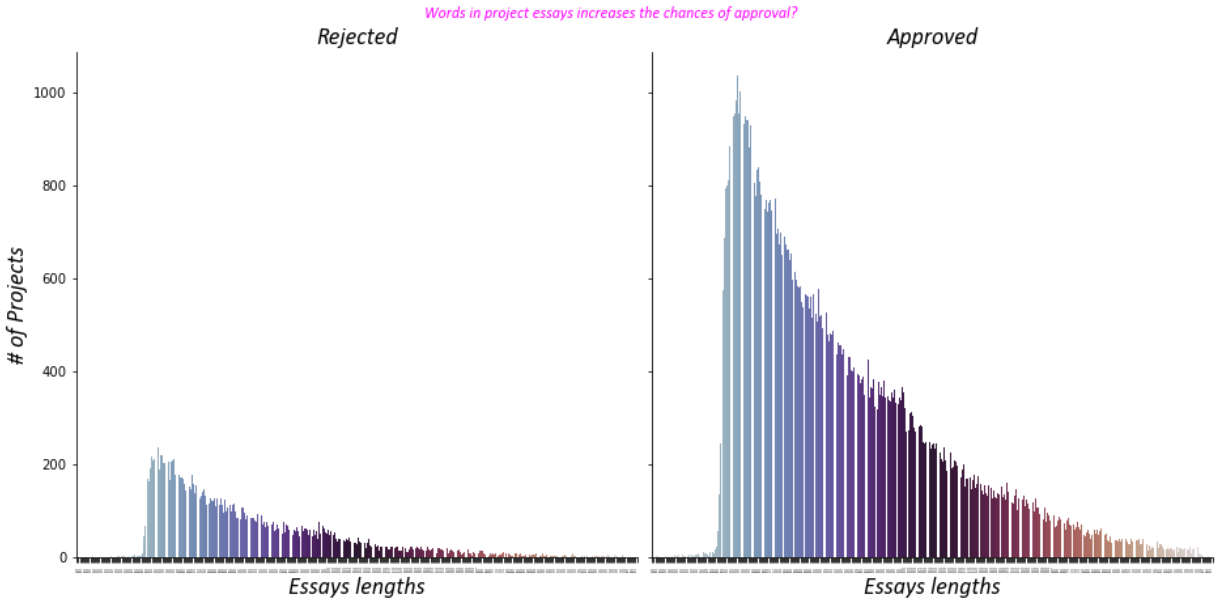
In [71]:
```
with plt.style.context('seaborn'):
    plt.figure(figsize=(15,6))
    sns.barplot(x='essays_lengths',y='num_submissions',data=proj_essay_len,palette='
    plt.xlabel('Words in projects essays',fontdict=lbl_dict)
    plt.ylabel('# of projects',fontdict=lbl_dict)
    plt.title('Projects submissions across various essay lengths',fontdict=ttl_dict)
    plt.xticks(size=2,style='oblique',rotation=90)
    plt.yticks(size=12,style='oblique')
```

In [72]:
```
dc_train_df['proj_essay_wrd_len'] = dc_train_df['essays'].str.split().apply(lambda r
```

In [73]:
```
proj_essay_apprv_status = pd.DataFrame(dc_train_df.groupby(['project_is_approved','p
                reset_index().rename(columns={'id':'submissions'})
```

In [74]:
```
projs_essays = sns.catplot(data=proj_essay_apprv_status,x='proj_essay_wrd_len',y='su
                        kind='bar',orient='v',palette='twilight',aspect=1,height=6.
axes = projs_essays.axes.flatten()
```

```
axes[0].set_title("Rejected",fontdict=lbl_dict)
axes[0].set_xlabel("Essays lengths",fontdict=lbl_dict)
axes[0].set_ylabel("# of Projects",fontdict=lbl_dict)
axes[1].set_title("Approved",fontdict=lbl_dict)
axes[1].set_xlabel("Essays lengths",fontdict=lbl_dict)
axes[1].set_ylabel("",fontdict=lbl_dict)
projs_essays.fig.suptitle("Words in project essays increases the chances of approval
projs_essays.set_xticklabels(rotation=90,size=2,style='oblique')
plt.tight_layout(pad=0.7,h_pad=0.3)
plt.show()
```



In [75]:
```
def feat_essay_buckets(val):
    """
    Description: This function is created for segregating the features values in buc

    Input: It accepts only one parameter:
        1. `val`: int
            Feature value used for allocating its bucket.

    Return: Bucket belongs to feature value.
    """
    if val >=0 and val<=120:
        return '0-120'
    elif val >=121 and val <=240:
        return '121-240'
    elif val >=241 and val <=360:
        return '241-360'
    elif val >=361 and val <=480:
        return '361-480'
    elif val >480:
        return '480+'
```

In [76]:
```
proj_essay_apprv_status['buckets'] = proj_essay_apprv_status['proj_essay_wrd_len'].a
```

In [77]:
```
essay_mul_idx = pd.MultiIndex.from_product([proj_essay_apprv_status['project_is_appr
                                            proj_essay_apprv_status['buckets'].uniqu

proj_essay_apprv_status = proj_essay_apprv_status.groupby(['project_is_approved','bu
.reindex(essay_mul_idx,fill_value=0).reset_index()

tot_essays_across_length_buckets = proj_essay_apprv_status.groupby(['buckets'])['sub

proj_essay_apprv_status['tot_subms'] = proj_essay_apprv_status['buckets'].apply(lamb
```

```
proj_essay_apprv_status['ratio_%'] = (proj_essay_apprv_status['submissions']*100.0)/
proj_essay_apprv_status
```

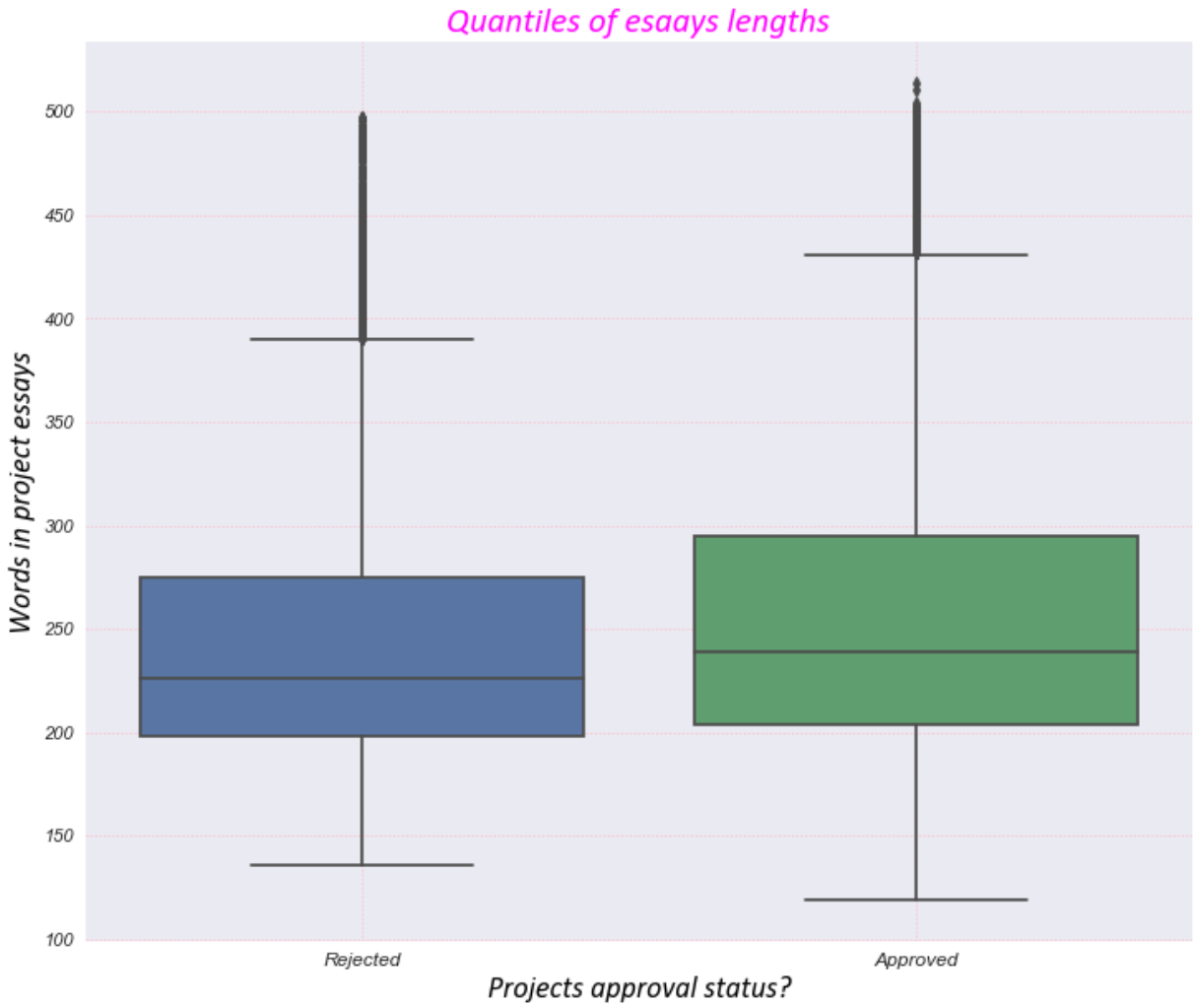
Out[77]:

	project_is_approved	buckets	submissions	tot_subms	ratio_%
0	0	121-240	9801	56736	17.274746
1	0	241-360	5767	43024	13.404147
2	0	361-480	930	9122	10.195133
3	0	480+	44	365	12.054795
4	0	0-120	0	1	0.000000
5	1	121-240	46935	56736	82.725254
6	1	241-360	37257	43024	86.595853
7	1	361-480	8192	9122	89.804867
8	1	480+	321	365	87.945205
9	1	0-120	1	1	100.000000

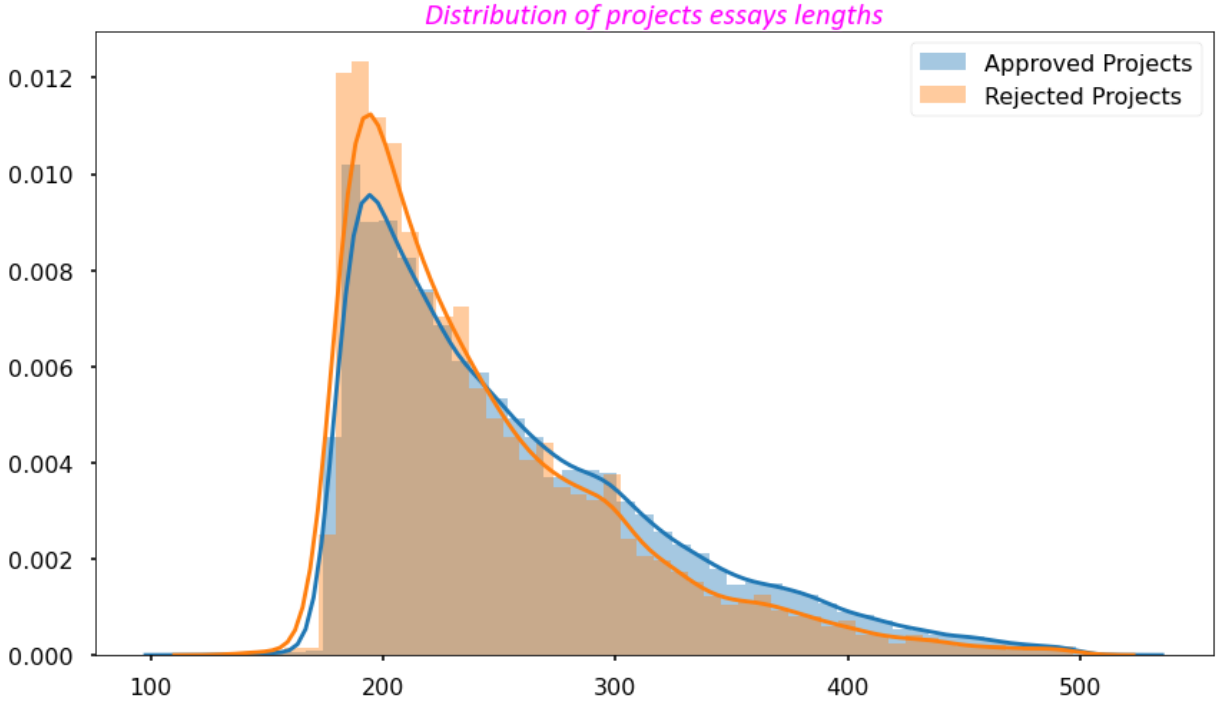
Here, we have similar acceptance and rejection ratio across all the buckets of essay lengths.

In [78]:

```
with plt.style.context('seaborn'):
    plt.figure(figsize=(12,10))
    sns.boxplot(data=[dc_train_df[dc_train_df['project_is_approved']==0]['proj_essay
                      dc_train_df[dc_train_df['project_is_approved']==1]['proj_essay
    plt.xlabel("Projects approval status?",fontdict=lbl_dict)
    plt.xticks([0,1],('Rejected','Approved'),style='oblique',size=12)
    plt.ylabel('Words in project essays',fontdict=lbl_dict)
    plt.yticks(style='oblique',size=11)
    plt.title('Quantiles of esaays lengths',fontdict=ttl_dict)
    plt.grid(which='major',linestyle=':',color='pink')
    plt.show()
```



In [79]:
```
with plt.style.context('seaborn-poster'):
    plt.figure(figsize=(14,8))
    sns.distplot(dc_train_df[dc_train_df['project_is_approved']==1]['proj_essay_wrd_
    sns.distplot(dc_train_df[dc_train_df['project_is_approved']==0]['proj_essay_wrd_
    plt.title("Distribution of projects essays lengths",fontdict=ttl_dict)
    plt.legend()
```



In the above distribution plot, we have a somewhat good difference

around the essay length with words 200. Otherwise, there is not much to seperate b/w the plots. So, we can say that essay length is not making the differentiation in the approval status.

In [80]:
```
apprv_projs_essay_len = dc_train_df[dc_train_df['project_is_approved']==1]['proj_ess
rej_projs_essay_len = dc_train_df[dc_train_df['project_is_approved']==0]['proj_essay
```

In [81]:
```
x = PrettyTable()
x.field_names = ["Percentile", "Approved Projects", "Rejected Projects", "Diff in pe

for i in range(0,101,5):
    x.add_row([i,
               np.round(np.percentile(apprv_projs_essay_len,i), 3),
               np.round(np.percentile(rej_projs_essay_len,i), 3),
               np.round(np.round(np.percentile(rej_projs_essay_len,i), 3) - np.round
print(x)
```

```
+------------+--------------------+--------------------+----------------------+
| Percentile | Approved Projects | Rejected Projects | Diff in percentiles |
+------------+--------------------+--------------------+----------------------+
|     0      |       119.0        |       136.0        |         17.0         |
|     5      |       184.0        |       182.0        |         -2.0         |
|     10     |       189.0        |       186.0        |         -3.0         |
|     15     |       194.0        |       190.0        |         -4.0         |
|     20     |       199.0        |       194.0        |         -5.0         |
|     25     |       204.0        |       198.0        |         -6.0         |
|     30     |       210.0        |       203.0        |         -7.0         |
|     35     |       217.0        |       208.0        |         -9.0         |
|     40     |       223.0        |       213.0        |        -10.0         |
|     45     |       231.0        |       219.0        |        -12.0         |
|     50     |       239.0        |       226.0        |        -13.0         |
|     55     |       248.0        |       234.0        |        -14.0         |
|     60     |       258.0        |       242.0        |        -16.0         |
|     65     |       269.0        |       251.0        |        -18.0         |
|     70     |       281.0        |       262.0        |        -19.0         |
|     75     |       295.0        |       275.0        |        -20.0         |
|     80     |       310.0        |       289.0        |        -21.0         |
|     85     |       329.0        |       304.0        |        -25.0         |
|     90     |       355.0        |       328.0        |        -27.0         |
|     95     |       392.0        |       369.0        |        -23.0         |
|    100     |       514.0        |       497.0        |        -17.0         |
+------------+--------------------+--------------------+----------------------+
```

Not many words differences in the essay lengths for approved or rejected projects.

Attribute-10

- **Cost per project**

In [82]:
```
dc_res_df.head(5)
```

Out[82]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95

In [83]:
```
proj_tot_price_quantty = dc_res_df.groupby('id').agg({'price':'sum', 'quantity':'sum
```

```
                        .rename(columns={'price':'tot_price','quantity':'tot_quantit

proj_desc_res_df = pd.merge(dc_train_df, proj_tot_price_quantty, on='id', how='left'
```

In [84]:
```
def feat_price_buckets(val):
    """
    Description: This function is created for segregating the features values in buc

    Input: It accepts only one parameter:
        1. `val`: int
            Feature value used for allocating its bucket.

    Return: Bucket belongs to feature value.
    """
    if val >=0 and val<=100:
        return '0-100'
    elif val >=101 and val <=200:
        return '101-200'
    elif val >=201 and val <=300:
        return '201-300'
    elif val >=301 and val <=400:
        return '301-400'
    elif val >=401 and val <=500:
        return '401-500'
    elif val >=501 and val <=600:
        return '501-600'
    elif val >=601 and val <=700:
        return '601-700'
    elif val >=701 and val <=800:
        return '701-800'
    elif val >=801 and val <=900:
        return '801-900'
    elif val >=901 and val <=1000:
        return '901-1000'
```

In [85]:
```
projects_prices = proj_desc_res_df[['project_is_approved','tot_price']].copy(deep=Tr
```

In [86]:
```
projects_prices['price_bucket'] = projects_prices['tot_price'].apply(lambda val: fea
projects_prices.head()
```

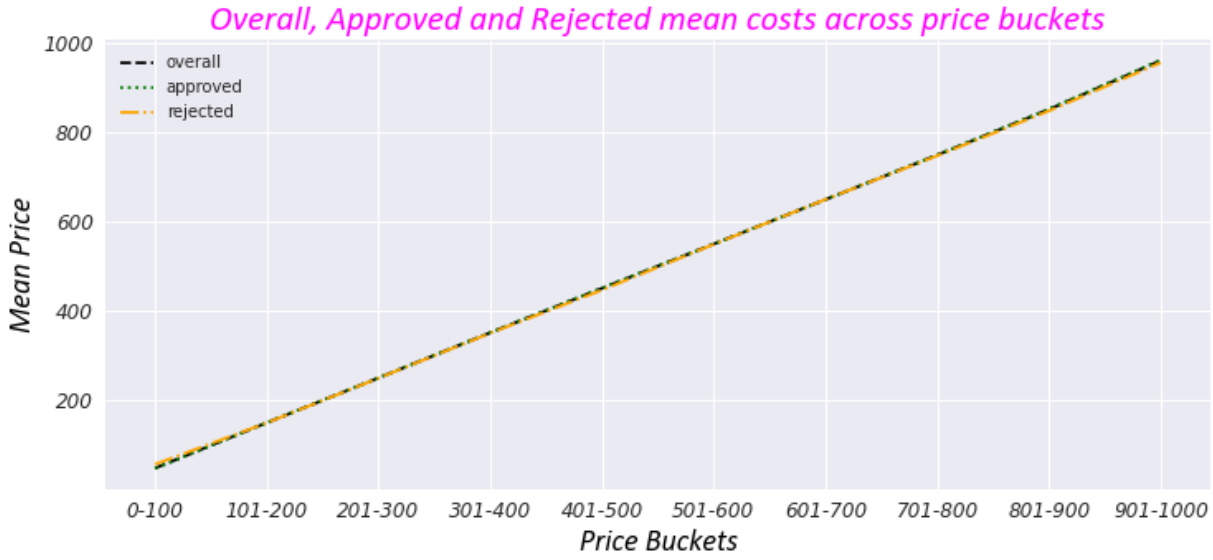
Out[86]:

	project_is_approved	tot_price	price_bucket
0	0	154.60	101-200
1	1	299.00	201-300
2	0	516.85	501-600
3	1	232.90	201-300
4	1	67.98	0-100

Mean project prices across buckets

In [87]:
```
overall_mean_projs_prices = projects_prices.groupby(['price_bucket'])['tot_price'].m
apprv_mean_projs_prices = projects_prices[projects_prices['project_is_approved']==1]
rej_mean_projs_prices = projects_prices[projects_prices['project_is_approved']==0].g

proj_price_analysis = pd.DataFrame({'price_bucket':overall_mean_projs_prices.keys(),
                                    'Overall_proj_mean_price':overall_mean_projs_pri
                                    'Apprv_proj_mean_price':apprv_mean_projs_prices.
                                    'Rej_proj_mean_price':rej_mean_projs_prices.valu
```

In [88]:
```
with plt.style.context('seaborn'):
    plt.figure(figsize=(12,5))
    line1 = sns.lineplot(data=proj_price_analysis,x='price_bucket',y='Overall_proj_m
    line1.lines[0].set_linestyle("--")
    line2 = sns.lineplot(data=proj_price_analysis,x='price_bucket',y='Apprv_proj_mea
    line2.lines[1].set_linestyle(":")
    line3 = sns.lineplot(data=proj_price_analysis,x='price_bucket',y='Rej_proj_mean_
    line3.lines[2].set_linestyle("-.")
    plt.xlabel("Price Buckets",fontdict=lbl_dict)
    plt.ylabel("Mean Price",fontdict=lbl_dict)
    plt.title("Overall, Approved and Rejected mean costs across price buckets",fontd
    plt.xticks(size=12,style='oblique')
    plt.yticks(size=12,style='oblique')
    plt.legend()
```



In [89]:
```
proj_price_analysis
```

Out[89]:

	price_bucket	Overall_proj_mean_price	Apprv_proj_mean_price	Rej_proj_mean_price
0	0-100	48.225090	47.292360	56.829375
1	101-200	149.937017	149.986648	149.627793
2	201-300	250.215771	250.314542	249.702848
3	301-400	351.530271	351.745039	350.490750
4	401-500	450.837879	451.693262	446.914814
5	501-600	550.041862	550.206356	549.362432
6	601-700	649.494857	649.410624	649.772025
7	701-800	749.202927	749.688760	747.583198
8	801-900	851.386549	852.394443	848.018874
9	901-1000	960.583769	961.898865	956.053992

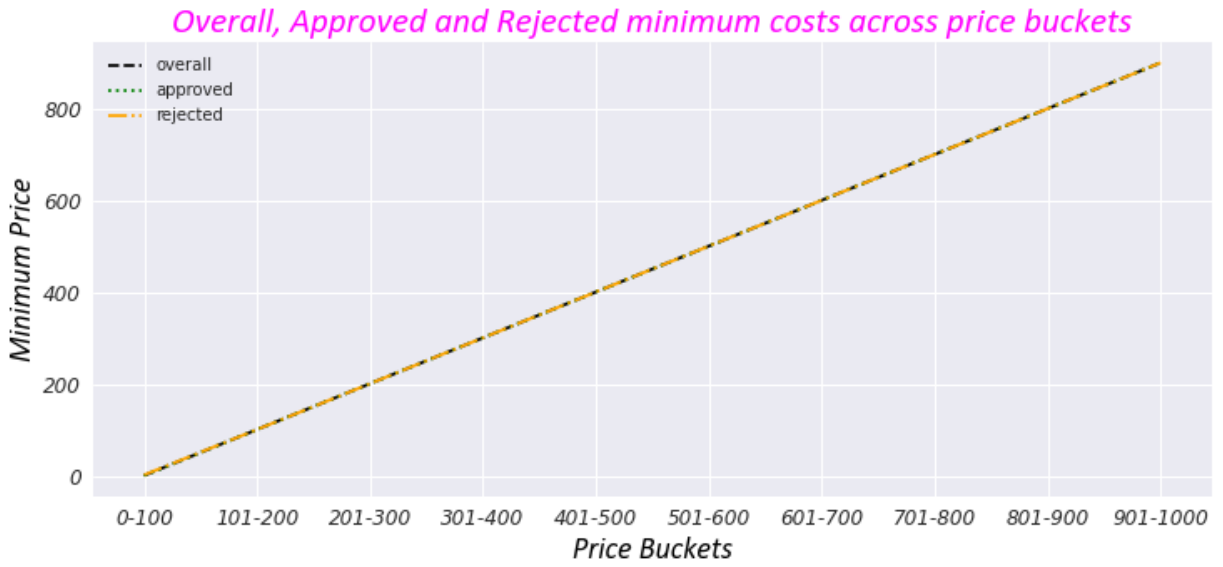
The above line plot and table shows us that the mean project price across the buckets is very much same.

Minimum project prices across buckets

In [90]:
```
overall_min_projs_prices = projects_prices.groupby(['price_bucket'])['tot_price'].mi
apprv_min_projs_prices = projects_prices[projects_prices['project_is_approved']==1].
rej_min_projs_prices = projects_prices[projects_prices['project_is_approved']==0].gr
```

```
proj_price_analysis = pd.DataFrame({'price_bucket':overall_min_projs_prices.keys(),
                                    'Overall_proj_min_price':overall_min_projs_price
                                    'Apprv_proj_min_price':apprv_min_projs_prices.va
                                    'Rej_proj_min_price':rej_min_projs_prices.values
```

In [91]:
```
with plt.style.context('seaborn'):
    plt.figure(figsize=(12,5))
    line1 = sns.lineplot(data=proj_price_analysis,x='price_bucket',y='Overall_proj_m
    line1.lines[0].set_linestyle("--")
    line2 = sns.lineplot(data=proj_price_analysis,x='price_bucket',y='Apprv_proj_min
    line2.lines[1].set_linestyle(":")
    line3 = sns.lineplot(data=proj_price_analysis,x='price_bucket',y='Rej_proj_min_p
    line3.lines[2].set_linestyle("-.")
    plt.xlabel("Price Buckets",fontdict=lbl_dict)
    plt.ylabel("Minimum Price",fontdict=lbl_dict)
    plt.title("Overall, Approved and Rejected minimum costs across price buckets",fo
    plt.xticks(size=12,style='oblique')
    plt.yticks(size=12,style='oblique')
    plt.legend()
```



In [92]:
```
proj_price_analysis
```

Out[92]:

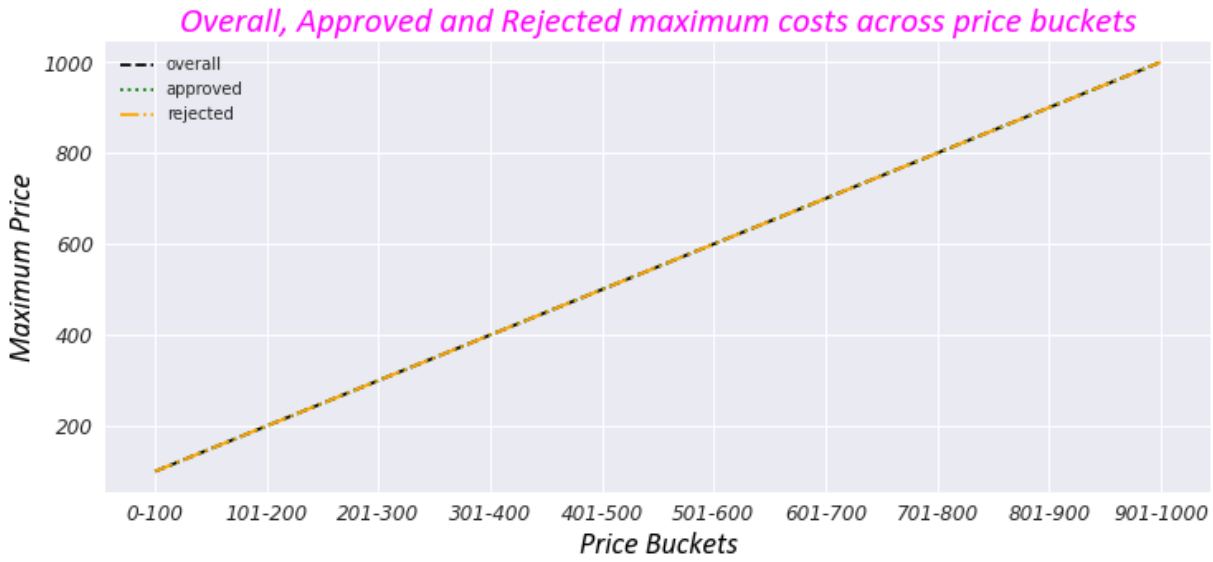
	price_bucket	Overall_proj_min_price	Apprv_proj_min_price	Rej_proj_min_price
0	0-100	0.66	0.66	1.97
1	101-200	101.00	101.00	101.00
2	201-300	201.01	201.01	201.06
3	301-400	301.00	301.00	301.05
4	401-500	401.00	401.00	401.06
5	501-600	501.02	501.02	501.15
6	601-700	601.03	601.03	601.08
7	701-800	701.11	701.12	701.11
8	801-900	801.03	801.08	801.03
9	901-1000	901.01	901.01	901.09

The above line plot and table shows us that the minimum project price across the buckets is very much same.

Maximum project prices across buckets

In [93]:
```
overall_max_projs_prices = projects_prices.groupby(['price_bucket'])['tot_price'].ma
apprv_max_projs_prices = projects_prices[projects_prices['project_is_approved']==1].
rej_max_projs_prices = projects_prices[projects_prices['project_is_approved']==0].gr

proj_price_analysis = pd.DataFrame({'price_bucket':overall_max_projs_prices.keys(),
                                    'Overall_proj_max_price':overall_max_projs_price
                                    'Apprv_proj_max_price':apprv_max_projs_prices.va
                                    'Rej_proj_max_price':rej_max_projs_prices.values
```

In [94]:
```
with plt.style.context('seaborn'):
    plt.figure(figsize=(12,5))
    line1 = sns.lineplot(data=proj_price_analysis,x='price_bucket',y='Overall_proj_m
    line1.lines[0].set_linestyle("--")
    line2 = sns.lineplot(data=proj_price_analysis,x='price_bucket',y='Apprv_proj_max
    line2.lines[1].set_linestyle(":")
    line3 = sns.lineplot(data=proj_price_analysis,x='price_bucket',y='Rej_proj_max_p
    line3.lines[2].set_linestyle("-.")
    plt.xlabel("Price Buckets",fontdict=lbl_dict)
    plt.ylabel("Maximum Price",fontdict=lbl_dict)
    plt.title("Overall, Approved and Rejected maximum costs across price buckets",fo
    plt.xticks(size=12,style='oblique')
    plt.yticks(size=12,style='oblique')
    plt.legend()
```



In [95]:
```
proj_price_analysis
```

Out[95]:

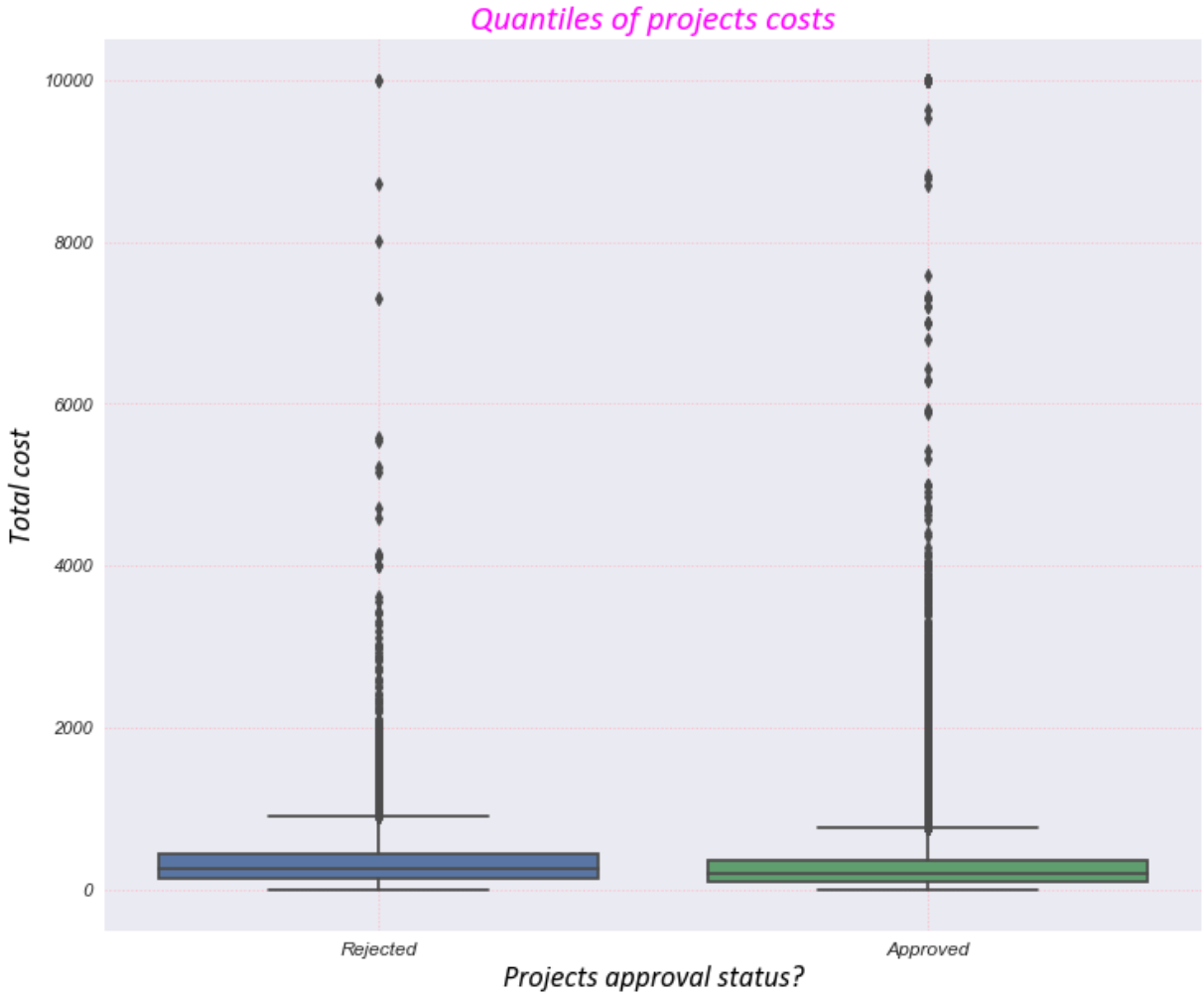
	price_bucket	Overall_proj_max_price	Apprv_proj_max_price	Rej_proj_max_price
0	0-100	100.00	100.00	100.00
1	101-200	200.00	200.00	200.00
2	201-300	300.00	300.00	300.00
3	301-400	400.00	400.00	399.99
4	401-500	499.99	499.99	499.99
5	501-600	600.00	599.99	600.00
6	601-700	700.00	700.00	699.99
7	701-800	800.00	800.00	799.99

	price_bucket	Overall_proj_max_price	Apprv_proj_max_price	Rej_proj_max_price
8	801-900	900.00	900.00	899.99
9	901-1000	1000.00	1000.00	999.99

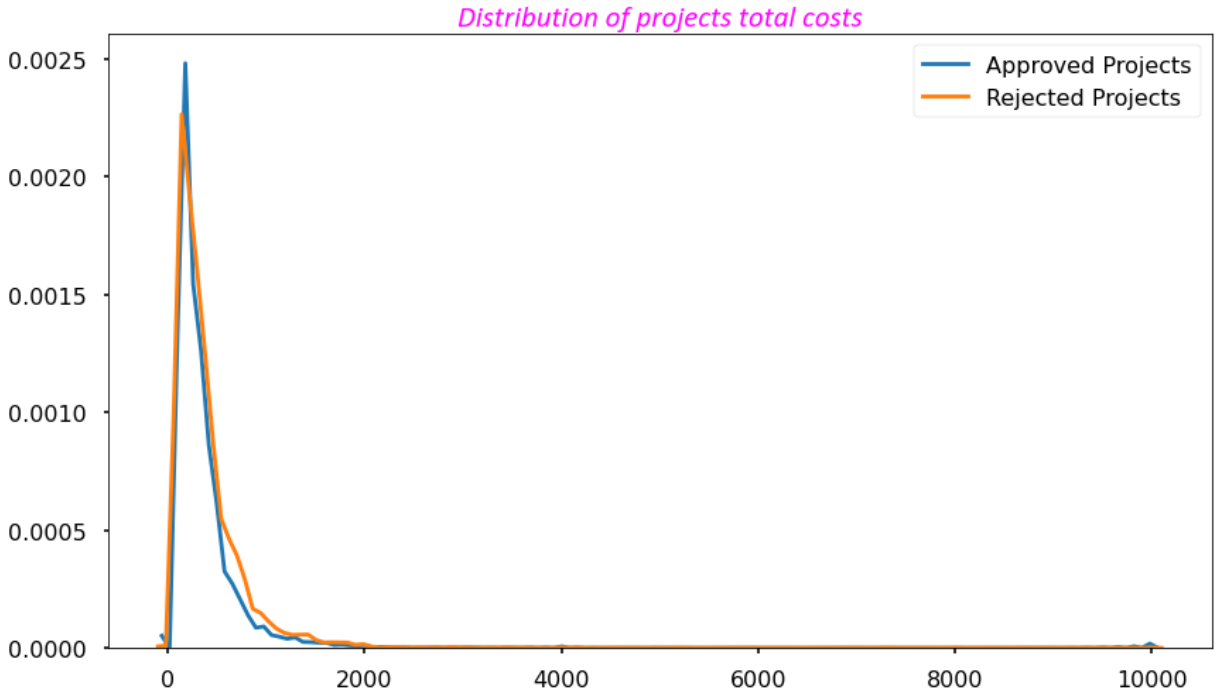
The above line plot and table shows us that the maximum project price across the buckets is very much same.

In [96]:
```
apprv_projs_costs = proj_desc_res_df[proj_desc_res_df['project_is_approved']==1]['to
rej_projs_costs = proj_desc_res_df[proj_desc_res_df['project_is_approved']==0]['tot_
```

In [97]:
```
with plt.style.context('seaborn'):
    plt.figure(figsize=(12,10))
    sns.boxplot(data=[rej_projs_costs,apprv_projs_costs])
    plt.xlabel("Projects approval status?",fontdict=lbl_dict)
    plt.xticks([0,1],('Rejected','Approved'),style='oblique',size=12)
    plt.ylabel('Total cost',fontdict=lbl_dict)
    plt.yticks(style='oblique',size=11)
    plt.title('Quantiles of projects costs',fontdict=ttl_dict)
    plt.grid(which='major',linestyle=':',color='pink')
    plt.show()
```



In [98]:
```
with plt.style.context('seaborn-poster'):
    plt.figure(figsize=(14,8))
    sns.distplot(apprv_projs_costs,hist=False,label="Approved Projects")
    sns.distplot(rej_projs_costs,hist=False,label="Rejected Projects")
    plt.title("Distribution of projects total costs",fontdict=ttl_dict)
    plt.legend()
```



Really nothing to separate here between the approved and rejected projects costs other than the difference at the first inflection point where rejected costs are a bit away from the approved costs.

In [99]:
```
x = PrettyTable()
x.field_names = ["Percentile", "Approved Projects", "Rejected Projects", "Diff in pe

for i in range(0,101,5):
    x.add_row([i,
               np.round(np.percentile(apprv_projs_costs,i), 3),
               np.round(np.percentile(rej_projs_costs,i), 3),
               np.round(np.round(np.percentile(rej_projs_costs,i), 3) - np.round(np.
print(x)
```

Percentile	Approved Projects	Rejected Projects	Diff in percentiles
0	0.66	1.97	1.31
5	13.59	41.9	28.31
10	33.88	73.67	39.79
15	58.0	99.109	41.109
20	77.38	118.56	41.18
25	99.95	140.892	40.942
30	116.68	162.23	45.55
35	137.232	184.014	46.782
40	157.0	208.632	51.632
45	178.265	235.106	56.841
50	198.99	263.145	64.155
55	223.99	292.61	68.62
60	255.63	325.144	69.514
65	285.412	362.39	76.978
70	321.225	399.99	78.765
75	366.075	449.945	83.87
80	411.67	519.282	107.612
85	479.0	618.276	139.276
90	593.11	739.356	146.246
95	801.598	992.486	190.888
100	9999.0	9999.0	0.0

The percentile table of project costs provides us some more details and we found that the percentile values of rejected prices are more than the approved prices. Thus, we can say that cost does plays an important role in acceptance or rejection.

Conclusion

- **If a teacher has previously submitted projects that doesn't really increases the approval chances of his/her current submission.**
  - **At the level of teacher also it is not making much of a difference.**
  - **We can exclude teacher-id and his/her previous submissions features as both of these are really not helping us in separating the classes.**
- **Numerical digits in the summary doesn't really ensures that project will be accepted.**
  - **No need of creating this new feature as it is not helping by any means.**
- **Essay lengths of accepted and rejected projects are quite similar in every aspect.**
  - **This new feature doesn't really needs to be included.**
- **Project cost does plays a crucial role in approval of proposal.**
  - **We definitely need to inlcude project cost in our features.**
- **Teacher prefix shows some variations among the categories thus we can look to keep it in our features.**
- **Every Grade category really holds the same percentage of acceptance and rejection, thus we look to skip this feature.**
- **We need to keep School state, subject category and subject sub-category in our features as there is good amount of variation.**